NAIST-IS-DD1361204

Doctoral Dissertation

Highly Reliable Memory Architectures Using Combination of In-Field Self-Repair, ECC and Aging Test

Gian Paolo Topico Mayuga

September 16, 2016

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Gian Paolo Topico Mayuga

Thesis Committee:

Professor Michiko Inoue	(Supervisor)
Professor Yasuhiko Nakashima	(Co-supervisor)
Visiting Professor Yasuo Sato	(Kyushu Institute of Technology)
Associate Professor Fukuhito Ooshita	(Co-supervisor)
Assistant Professor Yuta Yamato	(Co-supervisor)

Highly Reliable Memory Architectures Using Combination of In-Field Self-Repair, ECC and Aging Test*

Gian Paolo Topico Mayuga

Abstract

Embedded memory is occupying the most area in System-on-Chips (SoCs). It contributes to SoCs to have greater features, but at the expense of taking up the most area. The continuous scaling of nano-device technology makes memory more susceptible to errors. Aging-induced faults, as well as soft errors, manifest in field, which is a concern as causes of errors. Hence, reliability of memory is crucial to overall reliability of SoCs.

In-field test and repair, as well as Error Correction Codes (ECC), can be used to maintain reliability, and recently, these methods are used together to form a combined approach, wherein uncorrectable words are repaired, while correctable words are left to the ECC. In order to enhance reliability, a novel in-field repair strategy that repairs uncorrectable words, and possibly correctable words, for an ECC-based memory architecture is proposed. It executes an adaptive reconfiguration method that ensures fresh memory words are always used until spare words run out. To further enhance the reliability of memory, a more sophisticated test that can identify not only faulty memory cells, but aged ones as well, is used. A novel aging-aware in-field repair strategy that adaptively assigns aged and faulty words to functional repair or correction based on their vulnerabilities is proposed. Experimental results of both strategies demonstrate enhancement of reliability, and the hardware overhead contributions are small.

*Doctoral Dissertation, Graduate School of Information Science,
Nara Institute of Science and Technology, NAIST-IS-DD1361204, September 16, 2016.

Keywords:

memory repair, memory reliability, in-field test and repair, ECC, in-field repair strategy, remapping, aging-aware

Contents

List of Figures **vii**

1 Introduction **1**
1.1 Background 1
1.2 State of the Art 2
1.3 Thesis Contributions 4

2 Memory and Reliability **5**
2.1 Memory in Field 5
2.1.1 Memory Errors and Repair and ECC 5
2.2 Self-Repair and ECC 8
Correctable and Uncorrectable Words 8
Real Data of Errors in Memory 8
2.3 Combination of Self-Repair and ECC 9
2.4 Reliability Problem of Combination of Self-Repair and ECC in Field 10
2.5 Aging and Memory 11
2.5.1 Aging Test 13
2.6 Chapter Summary 13

3 Reliability Enhancement Using Combination of In-Field Self-Repair and ECC **15**
3.1 Proposed In-Field Repair Strategy 15
3.2 Memory Architecture Using Combination of In-Field Self-Repair and ECC 16
3.2.1 Architecture Details 17
User Mode 17
Test Mode 18

3.2.2 Memory Elements 20
During User Mode 20
During Test Mode 20
3.3 Proposed Remap Controller Implementation 20
3.3.1 Reconfiguring the Remap CAM 21
3.3.2 Remap CAM Operations 22
3.4 Reliability Model Using Combination of In-Field Self-Repair and ECC 24
3.4.1 Probability of Words 25
Probability due to Hard Error 25
Probability due to Soft Errors 26
Probability due to Scrubbing 26
3.4.2 Reliability of Word for 1st Self-Test Period 27
3.4.3 Reliability of Memory using Traditional Strategy 27
3.4.4 Reliability of Memory for 1st Self-Test Period 27
3.4.5 Reliability of Word during One Test Period 27
3.4.6 Reliability of Self-Test Period 28
3.4.7 Reliability of Memory System using Proposed Strategy . . 29
3.5 Experimental Results 29
3.5.1 Reliability Evaluation 29
3.5.2 Hardware Overhead 35
Area Overhead 35
Power and Latency Overhead 36
3.6 Chapter Summary 38

4 Reliability Enhancement Using Combination of Aging-aware Adaptive In-Field Self-Repair and ECC 39
4.1 Aging and Memory Cells and Memory Words 39
4.1.1 State Model of Memory Cells 40
4.1.2 Probability of Memory Cell States 40
4.1.3 Word Status Types 41
4.1.4 Probability of Memory Word Status Types 42

4.2 Memory Architecture Using Combination of Aging-Aware Adaptive In-Field Self-Repair and ECC . . . 43
4.2.1 Extended In-Field Repair Strategy . . . 43
4.2.2 Architecture Details . . . 44
4.2.3 Extended Remap Controller . . . 45
4.3 Reliability Model Using Combination of Aging-aware Adaptive In-Field Self-Repair and ECC . . . 46
4.3.1 Probability at Cell-Level (Bit-Level) . . . 48
State Probability of Cell . . . 48
Transition Probability of Cell . . . 48
Probability of Soft Error in a Cell . . . 49
4.3.2 Probability at Word-Level . . . 49
Probability of Word . . . 49
Transition Probability of Word . . . 50
Probability of Soft Error in a Word . . . 51
4.3.3 Reliability of a Memory Word . . . 51
4.3.4 Reliability of Self-Test Period . . . 52
4.3.5 Reliability of Memory System . . . 53
4.4 Experimental Results . . . 54
4.4.1 Reliability Evaluation . . . 54
4.4.2 Hardware Overhead . . . 56
Area Overhead . . . 56
Power and Latency Overhead . . . 57
4.5 Chapter Summary . . . 57

5 Support for Memory of Higher Technology 59
5.1 Memory with Burst Rate Operation . . . 59
5.2 Latency Optimization . . . 60
5.3 Large Memory Block Size (up to 512 bits) and ECC Protection Strength . . . 61
5.4 Implementation of Proposed Strategy to Cache Memory . . . 64
5.5 Summary . . . 65

6 Conclusion **66**
6.1 Memory Architectures for Reliability Enhancement 66
6.2 Summary 67

References **68**

Publication List **72**

List of Figures

1.1 SoC Area Partitioning [9] 2
1.2 Time when memory test is applied 3

2.1 Repair and ECC for Hard Errors a) faulty row must be replaced by spare row b) faulty column must be replaced by spare column c) row with single error replaced by spare row d) column with single error replaced by column row e) erroneous code word replaced by spare code word f) code word with single error can be corrected with ECC instead to increase repair efficiency 6
2.2 Functional Repair for Memory Words 7
2.3 Correction for Memory Words 8
2.4 Traditional Combination Strategy of Repair and ECC a) Uncorrectable word is repaired using spare word b) Correctable word is left to be corrected by ECC 9
2.5 Reliability Problem of Combination of Repair and ECC in Field . 11
2.6 Memory Cell Operating Level and Aging a) Memory cell only need to be at the minimum (MIN) operating level for it to be turned 'ON' b) Aging affects the memory cell such that the new MIN operates at a higher level, and if the operating level lower than the new MIN is applied, the memory cell is unsure if it is 'ON' or 'OFF' c) Memory cells can be identified as aged once the MIN operating level passes the AGED level 12
2.7 On Chip Aging Sensor Circuit [17] 14

3.1 Proposed In-Field Strategy a) Repair word with 1 faulty bit if spare is available b) ECC can handle, if no more spares 16
3.2 Proposed ECC-Based Memory Architecture 17

3.3 Data Flow for User Mode 18
3.4 Data Flow for Test Mode 19
3.5 Writing New Entry when Remap CAM is Not Full 21
3.6 Writing New Entry when Remap CAM is Full 22
3.7 Self-test and repair and scrubbing 25
3.8 Reliability of proposed strategy vs traditional strategy for various memory sizes 31
3.9 Reliability of proposed strategy vs traditional strategy for various spare word sizes 31
3.10 Reliability of proposed strategy vs traditional strategy for varying hard error rate given soft error rate λ_s=10^{-7} 32
3.11 Reliability of proposed strategy vs traditional strategy for varying soft error rate given hard error rate λ_h=10^{-11} 33
3.12 Reliability of proposed strategy vs traditional strategy for various word lengths up to 32 bits 33
3.13 Reliability of proposed strategy vs traditional strategy for various word lengths up to 512 bits 34
3.14 Reliability of proposed strategy vs traditional strategy for varying self-test and repair periods 34

4.1 2-State Model vs. 3-State Model for Memory Cell 40
4.2 Word Status Types Based on Priority For Repair 42
4.3 Extended In-Field Repair Strategy 44
4.4 Proposed ECC-Based Aging-Aware Memory Architecture 45
4.5 Remap CAM Using Extended In-Field Repair Strategy 46
4.6 Writing New Entry when Extended Remap CAM is Not Full . . . 47
4.7 Writing New Entry when Extended Remap CAM is Not Full . . . 48
4.8 Reliability of extended proposed strategy vs previous vs traditional strategy for error rate of λ=10^{-11} 55
4.9 Reliability of extended proposed strategy vs previous vs traditional strategy for error rate of λ=10^{-10} 55
4.10 Reliability of extended proposed strategy vs previous vs traditional strategy for error rate of λ=10^{-9} 56

5.1 Block status for memory with burst rate operation 59
5.2 Remap CAM Access During User Mode 60
5.3 Modified architecture to optimize performance 61
5.4 ECC for 512bit Block Size at Block and Word Level 62
5.5 Reliability for 512bit Block Size Memory with ECC at Word Level 63
5.6 Direct-mapped Cache Memory 64
5.7 Proposed Repair Integrated into Cache 65

1 Introduction

1.1 Background

Embedded memory is used to integrate with logic partitions located on the same chip in order to address requirements of high performance and low power. Memories have emerged to predominantly occupy the most silicon area in System-on-Chips (SoC) as shown in Figure 1.1. This integration of logic and memory, with the continuous scaling of the device sizes, leads to a more powerful SoC, which works on the latest applications that require larger memory. However, it introduces drawbacks on manufacturing yield and field reliability. Since memory occupies most of the SoC, the memory yield directly affects the overall yield of an SoC [1]. To improve the memory yield, repair by using redundancy has traditionally been used, and recently has been combined with error correction codes (ECC) [2]. Also, post manufacturing reliability failures might highly occur for nanoscale devices, and this challenge may be resolved by using periodic field-level repair [1]. Several works address the effect of increasing failure rates and the importance of resolving the challenges it brings for both yield and reliability [2–6].

At the same time, the continuous scaling of nanodevice technology produces random process variation which causes time-dependent deviation of memory cell parameter, and brings aging which results in significant degradation of Static Noise Margin (SNM) [7, 8]. Aging is due to Negative-Bias Temperature Instability (NBTI), Time Dependent Dielectric Breakdown (TDDB) and Hot Carrier Degradation (HCI), which causes memory cell data flipping, and increase in cell access time. Continuous monitoring of the memory cell, as well as actively using spare elements via performing in-field self-repair, is necessary to mitigate the effects of aging.

Post manufacturing reliability failures and aging are identified as factors that

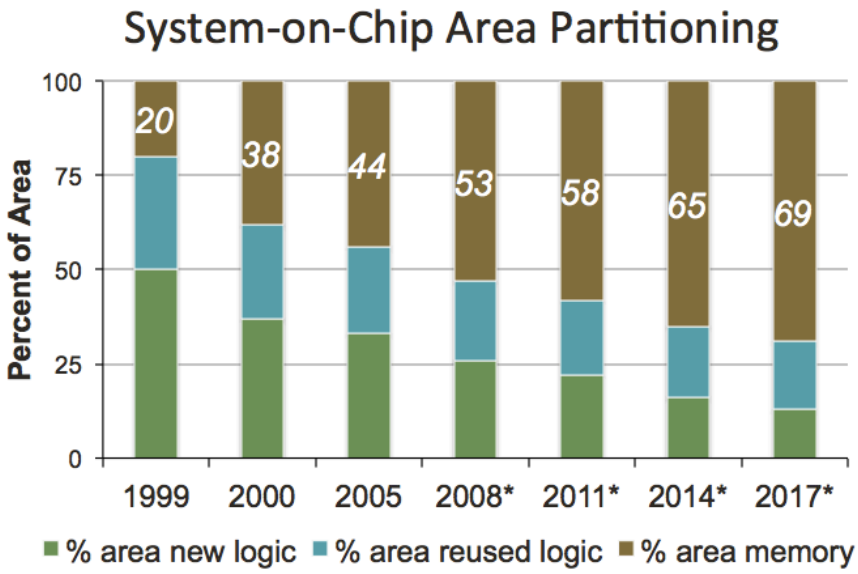

Figure 1.1: SoC Area Partitioning [9]

affect reliability of memory, and thus, the overall reliability of SoCs. Periodic field-level test and repair and ECC are used as solutions to maintain reliability.

1.2 State of the Art

Memory test is an important point of interest in the industry today. Memory historically advances faster in terms of technology than microprocessors. This is due to the fact that memory is generally less complex, and is mostly composed of arrays of information-storing devices that are designed for either speed or density. This implies that the test and repair requirement must be at par with the advancement of memory technology. Memory test is applied at 1) during manufacturing or before memory is shipped out, or 2) while in-field when memory is already in operation, as shown in Figure 1.2. Manufacturing test is a general procedure all integrated chips go through before shipping, and also during this time, repair is applied to replace faulty elements in order to improve the yield. Combination of repair and ECC has been applied during manufacturing test and has shown to be effective in enhancing yield. Also, due to the aforementioned impending issues regarding reliability, the importance of memory, and SoCs as a whole, to

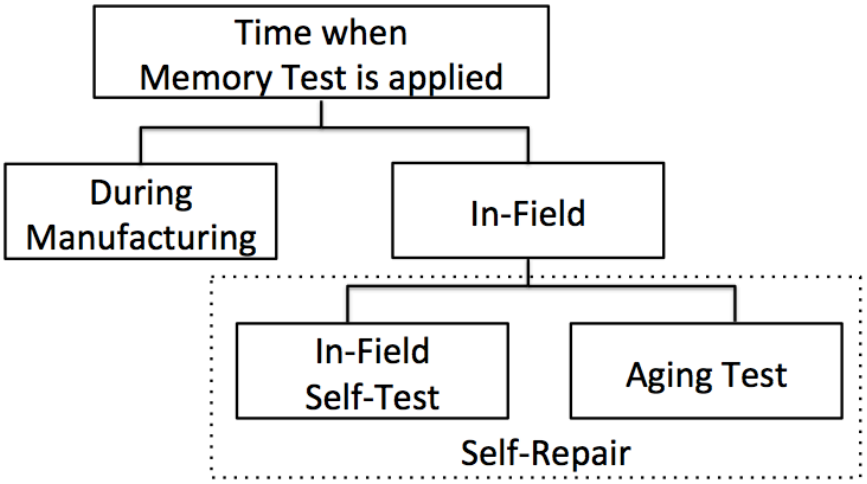

Figure 1.2: Time when memory test is applied

remain in operation while in field is an immediate priority by researchers in the design and test community, and is being explored further. The configuration for combination of repair and ECC can be used for reliability enhancement as well, and it is explored in this work.

Also falling under the in-field category is aging test. Various researches on on-chip sensor design have been thoroughly explored in order to detect aging in memory cells as it can induce faults, and thus failures, to occur. Recent industry tools are now exploring embedding aging sensors into memory systems [10]. The capability to detect aging in memory cells can be used for reliability enhancement, and it is also explored in this work.

The reliability problem presented by combining repair and ECC in field is explained in Chapter 2, and a solution is presented in Chapter 3. At the same time, we present the importance of aging and its effect in memory, and by using the same configuration of repair and ECC, we enhance reliability further in Chapter 4. Overall, this work outlines the importance of fault-tolerant operations while in field, and the identified reliability problems are presented with solutions.

1.3 Thesis Contributions

The contributions of this thesis are the following:

- A novel adaptive remapping configuration architecture that enhances reliability by repairing uncorrectable words first, and repairing correctable faulty words when spare space is remaining is proposed. To guarantee remapping ability of uncorrectable words, the remapping of correctable faulty words is cancelled if necessary.

- A stochastic reliability model of the proposed architecture, where reliability improvements by memory scrubbing and repair of correctable faulty words are evaluated as well as that of uncorrectable word repair is proposed. The simulation-based evaluation shows that the proposed architecture provides greater reliability than the traditional architecture.

- The proposed remap controller that executes an adaptive reconfiguration method to realize the in-field repair strategy is implemented. The overhead of the remap controller is found to be small.

- The 2-state model used in the proposed architecture is extended into a 3-state model for memory cells, where memory cells are classified to be healthy, aged or faulty. The stochastic evaluation is presented, and the further enhancement of reliability is shown.

- Given the proposal of 3-state model for memory cells, a novel aging-aware architecture that enhances reliability further by repairing faulty words and aged words based on their vulnerabilities is proposed. Memory words are classified into 5 status types based on the 3-state model of memory cell. To ensure repair of a more vulnerable word, the previous repair of a less vulnerable word is cancelled if necessary.

- The proposed remap controller is extended such that it adaptively reconfigures a remap CAM based on status types of the memory words based from the 3-state model.

2 Memory and Reliability

In this chapter, the importance of ensuring that memory is reliable in its lifetime is discussed. The vulnerabilities encountered by embedded memory in SoCs, the efficiency of using the combination of repair and ECC for memory, and the reliability problem it presents are presented. Also, the effect of aging in memory is introduced and how incorporating aging-awareness in memory systems has been recently prioritized.

2.1 Memory in Field

Memory occupies the most area in SoCs, which means memory also has the most probability of encountering an error. Therefore, we have to consider that occurrences of errors, and eventually failures, is an important aspect in maintaining reliability, especially after the memory devices are already used in field.

2.1.1 Memory Errors and Repair and ECC

Memory errors are classified as either hard or soft errors [3]. Memory errors can either be repaired (hard errors) or corrected (hard and soft errors). Hard errors are errors caused by permanent hardware faults. The causes are 1) single bit fault for a cell in a memory word, 2) faults for all cells in a memory row or column, or 3) faults for all cells in a memory array. While faults that appear on rows or columns and arrays are most of the time repaired right after manufacturing test and are wholly replaced, single bit faults on a cell represent issues in allocation of spares. As an example, as shown in Figure 2.1(a)-(b), the faulty row or column is replaced by a spare or column. However, for a row or column with single bit faults as shown in Figure 2.1(c)-(d), replacements can only be a spare row or column. The priority is to repair as much as possible since it is important to

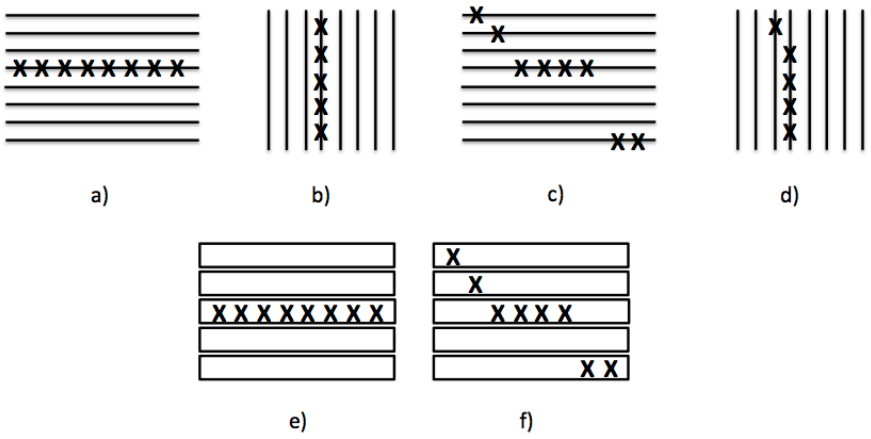

Figure 2.1: Repair and ECC for Hard Errors a) faulty row must be replaced by spare row b) faulty column must be replaced by spare column c) row with single error replaced by spare row d) column with single error replaced by column row e) erroneous code word replaced by spare code word f) code word with single error can be corrected with ECC instead to increase repair efficiency

have a good yield. There will be remaining spares that are left over, but they are only limited if most of the spares have already been used. Careful allocation of these spares becomes important for in-field repair.

Memory repair is usually done by external means like using fuses or other methods [3], usually during manufacturing test. However, for embedded memories, especially being in operation in field, physical reconfigurations are costly. Functional reconfiguration techniques are instead used where memory access is logically re-assigned to a spare instead of the faulty row/column [11], word [4] or cell [12]. The work [12] proposed performing repair through the use of a content addressable memory and a functional remapping occurs in the cell level if the original cell is faulty. The work [5] proposed performing repair through the use of a content addressable memory and a functional remapping occurs in the word level as shown in Figure 2.2. This work focuses on a word-based organization, and functional word repair will be discussed further later on.

Soft errors are caused by transient phenomena such as alpha rays or cosmic rays that change the data stored in memory cells. Solutions include making the

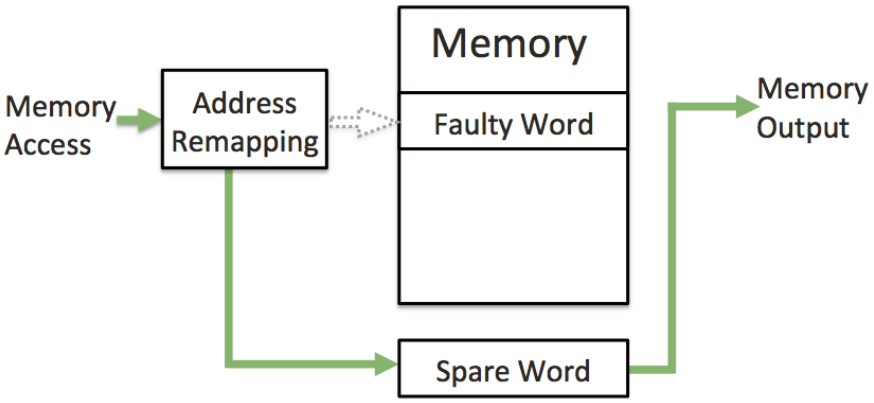

Figure 2.2: Functional Repair for Memory Words

memory cell design more radiation-hardened in the device level layer. Another is to integrate fault-tolerant solutions on the memory system-level by using ECC. ECC is already featured in almost all IC implementations such as prominent Intel processors, and other leading IC companies. Correction capability depends on the number of correctable bits by the ECC. Single-Error Correction (SEC) is the most common type. Other types are Single-Error Correction Double-Error Detection (SECDED), which is in addition to SEC, it can detect 2 errors in a codeword, Double-Error Correction (DEC), which can correct up to 2 errors in a codeword, and Chipkill, which can correct up to 4 errors in a codeword. ECC is effective for correcting both hard and soft errors, but at the expense of extra bits added to the memory data to form a codeword, as well as encoder and decoder circuits to generate and extract the codeword. The higher the number of bits that ECC can correct, the greater the overhead. As an additional protection, the error-free data output can be fed back to the source memory word and can be re-written, in order to ensure the integrity of the memory data. When the error-free data is re-written periodically, this process is called scrubbing, and it can either be done via software or hardware controller [13]. The use of ECC and scrubbing is shown in Figure 2.3.

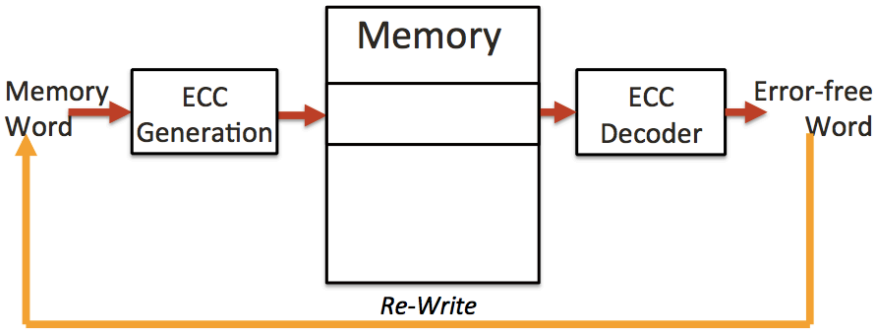

Figure 2.3: Correction for Memory Words

2.2 Self-Repair and ECC

Given the use of ECC in a memory system, the memory is now organized as a collection of codewords. Given a faulty codeword, a spare codeword is used to perform repair as shown in Figure 2.1(e). Also, given a codeword with a single error as shown in Figure 2.1(f), a whole spare codeword does not have to be allotted to perform repair since ECC can be used instead in order to provide error-free operation in memory.

Correctable and Uncorrectable Words

Given that ECC handles memory errors in memory words in field, this allows re-classification of memory words and errors. Erroneous memory words are classified as either 1) correctable words, which can be corrected by ECC, and 2) uncorrectable words, which the ECC cannot handle. We can then refer to a memory error as either a 1) correctable error, if it is in a correctable word, and 2) uncorrectable error, if it is in an uncorrectable word. Handling of correctable and uncorrectable words is important since it affects memory lifetime.

Real Data of Errors in Memory

A recent study that explores a large experimental scale of errors in DRAM memory used by Google servers has been reported [14]. The large gathered data shows a practical picture of how errors affect memory lifetime. The crucial conclusions are as follows: 1) A memory word with correctable error is more likely to develop

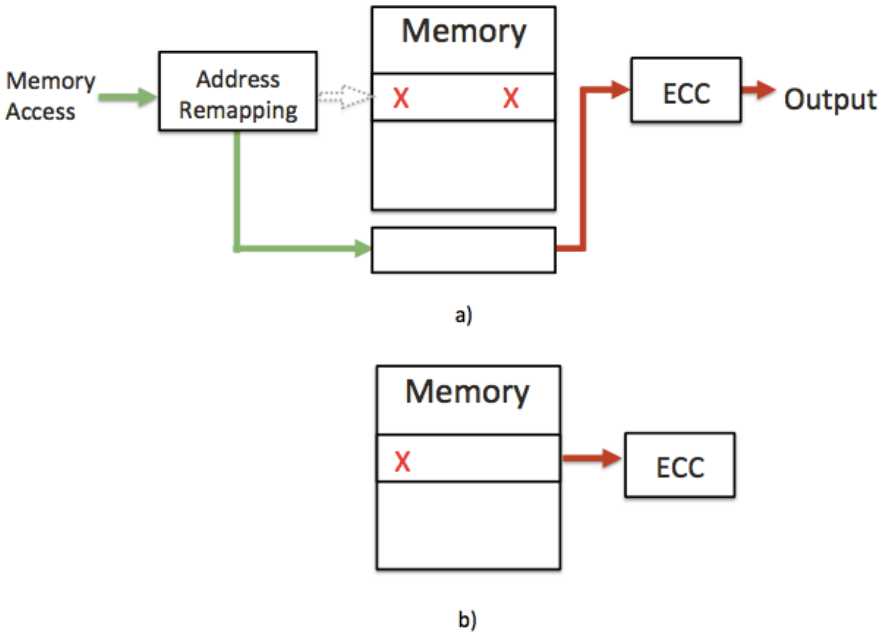

Figure 2.4: Traditional Combination Strategy of Repair and ECC a) Uncorrectable word is repaired using spare word b) Correctable word is left to be corrected by ECC

uncorrectable errors (In the same month, there is a 70-80% chance that an uncorrectable error will develop after a correctable error appears in the same Dual Inline Memory Module (DIMM)); 2) Error rates are unlikely to be dominated by soft errors, and hard errors contribute the most for memory failures, and 1.3% of machines across the entire fleet are affected by uncorrectable errors per year, in such a way that even if a DIMM is replaced with a new one at the first sign of an uncorrectable error, another uncorrectable error appears on a different DIMM. These observations on module-based memory devices indicate that it is important to mitigate the effect of uncorrectable errors in the memory.

2.3 Combination of Self-Repair and ECC

The combination of self-repair and ECC synergistically improves both yield and reliability when it is systematically used into a combination strategy as shown in

Figure 2.4. Using single error correction, the combination strategy is as follows:

- Repair a word if it has 2 or more faulty bits.
- Leave it to be corrected by ECC if it has 1 faulty bit.

The combination strategy, which is referred to as the traditional combination strategy in this work, has been adapted in several works [2–6]. The methods introduced in [2–4] improve yield by repairing uncorrectable words given a limited amount of spare space. They repair uncorrectable faulty words and leave correctable faulty words in charge to ECC. As a result, memory yield can be improved much, even if defect density is large. This is because memory will work well in cooperation with ECC if spare space cannot cover all the faulty words. The work [5] studies online periodic self-test using a transparent BIST [15]. The transparent BIST is enhanced so that it can distinguish between correctable and uncorrectable errors and it is used to repair uncorrectable words. The work [6] also integrates in-field self-test and ECC, where ECC is used to distinguish hard and soft errors, and all the identified hard errors are repaired.

All in all, this strategy ensures that spares are only allotted for words that correction cannot handle. It improves repair efficiency and ensures that uncorrectable words are repaired.

2.4 Reliability Problem of Combination of Self-Repair and ECC in Field

Some issues arise from using combination of self-repair and ECC solution in field. In [2–4], since correctable faulty words are shipped along with the memory chip, correctable errors are already in the memory system, and there is a high probability that these correctable errors will develop into uncorrectable errors, which affects reliability, and shortens memory lifetime. In [5], memory words with correctable errors are left active in the memory system, and are vulnerable to uncorrectable errors. In [6], the available spare space might quickly run out, and the next occurrence of an uncorrectable error causes failure, and thus shortens memory lifetime. For all these cases, they present a reliability problem since

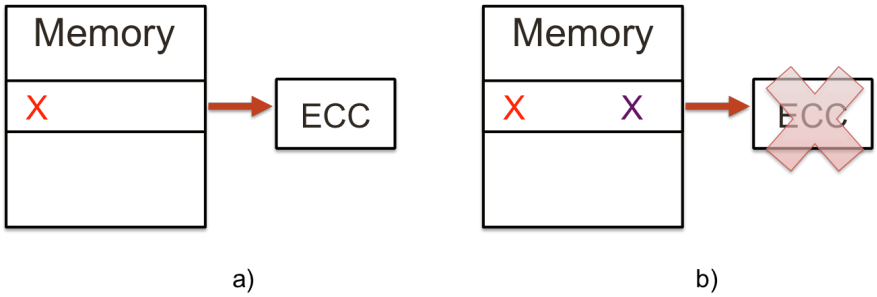

Figure 2.5: Reliability Problem of Combination of Repair and ECC in Field

the correctable word is part of the memory system as shown in Figure 2.5(a). The moment the correctable word becomes an uncorrectable words as shown in Figure 2.5(b), failure may occur.

2.5 Aging and Memory

Aging-induced faults manifest in field and it is an important cause for concern. Memory cells affected by NBTI suffer from SNM degradation. In [16], SNM degrades by about 8% after 3 years on 100nm and 70nm cells under continuous stress. In [17] which features 6T SRAM cells in 65nm technology, the SNM of aged cell is reduced by 12.5% with respect to a fresh cell after 6 years. Thus, memory cells become prone to errors. Memory cells suffer from bit flips during read operations, and memory read failures.

The effect of SNM degradation is shown in Figure 2.6. For a 'fresh' memory cell shown in Figure 2.6(a), the minimum (MIN) value is enough to turn on ('ON') the memory cell. Aging causes the MIN value to become degraded as shown in Figure 2.6(b). This implies that a greater value than the previous MIN value is needed to put the memory cell in 'ON' state. A problem occurs when the same operating level is used all throughout to turn on the memory cell. For example, the operating level of the cell is just right above the 'MIN' level, and when degradation occurs, the 'MIN' level changes, and failure may occur. Thus, having an aging sensor can inform the system about the status of the memory cell as aged, as shown in Figure 2.6(c).

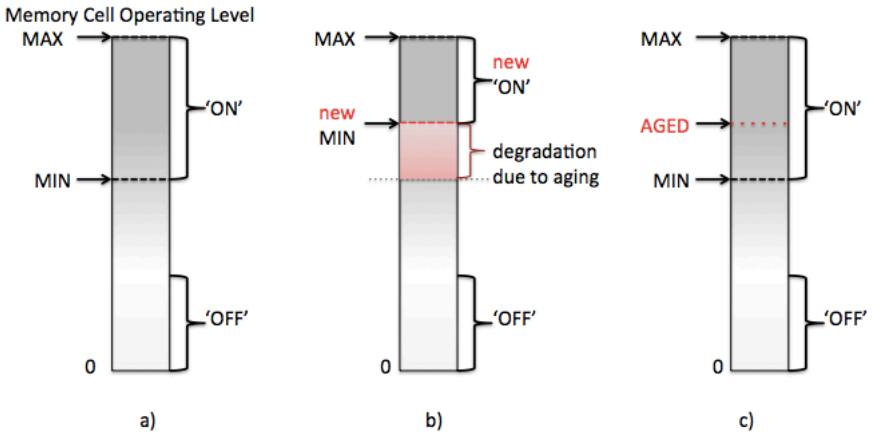

Figure 2.6: Memory Cell Operating Level and Aging a) Memory cell only need to be at the minimum (MIN) operating level for it to be turned 'ON' b) Aging affects the memory cell such that the new MIN operates at a higher level, and if the operating level lower than the new MIN is applied, the memory cell is unsure if it is 'ON' or 'OFF' c) Memory cells can be identified as aged once the MIN operating level passes the AGED level

2.5.1 Aging Test

Awareness and identification of the agedness of memory cells have been recently investigated. A self-test circuit to detect SRAM cell aging was proposed, called on-chip aging sensor (OCAS) [17], which enables degraded SNM cells to be identified as aged cells. The block diagram of this self-test circuit is shown in Figure 2.7. During normal operation, VDD is connected to the memory cells. Also at this time, the aging circuit is completely disconnected from VDD, as it is gated off from power supply. This ensures that the OCAS remains unaffected by aging. During testing operation, the memory cells are disconnected from VDD. A memory cell under test is chosen among the memory cells. Two consecutive write operations are performed to the cell under test, first the opposite of the previously written value, and then the original value. This causes a discharge to the value at the point of VDD'. This value is compared to that of a reference memory cell that does not experience aging. If VDD' is not discharged well, aging is detected.

The aging detection is also explained in terms of operating level. If the operating level of the cell under test is less than the reference memory cell (greater discharge), then the memory cell is non-aged. If the operating level of the memory cell under test is greater than the reference memory cell (less discharge), then the memory cell is aged. The aging test is performed for each memory cell.

Also, active use of spare elements has been shown to effectively mitigate the effects of aging [18]. Combining the capability of identifying aged elements and an adaptive way of reconfiguring spare elements, an aging-aware architecture can now be formulated in order to further enhance reliability.

2.6 Chapter Summary

Given the vulnerabilities encountered by embedded memory in SoCs, combination of self-repair and ECC can be configured such that memory repair is performed efficiently. The reliability problem that can occur from this configuration has been presented. The solution for this problem is discussed in Chapter 3. Moreover, aging that can make memory more susceptible to failure has been explained. The solution to make the memory architecture become aging-aware is presented in

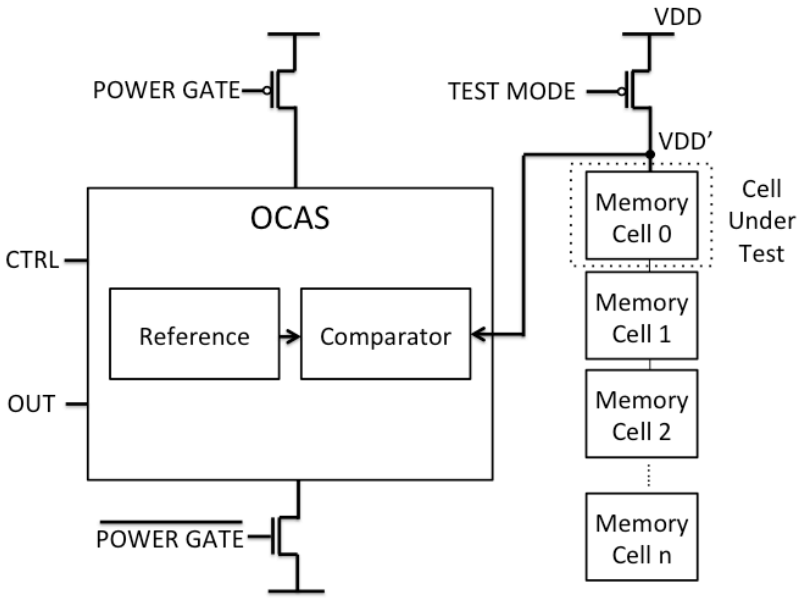

Figure 2.7: On Chip Aging Sensor Circuit [17]

Chapter 4.

3 Reliability Enhancement Using Combination of In-Field Self-Repair and ECC

While the combination of repair and ECC improves the yield, it poses a reliability problem for the memory in field. In this chapter, a solution to the reliability problem of the combination of repair and ECC is presented. A memory architecture that enhances reliability by using an in-field repair strategy that repairs uncorrectable words, as well as correctable words, if spare space is available is described in detail.

3.1 Proposed In-Field Repair Strategy

In the proposed strategy, given enough spares, both uncorrectable and correctable words are repaired using functional remapping after applying self-test and repair. It improves reliability since the possibility of uncorrectable words during system use is reduced. However, the number of uncorrectable words may increase, and not all erroneous words may be remapped to fresh spare words. In such a case, priority is given to uncorrectable words to be remapped, to the extent of cancelling previously remapped correctable words. The correctable words are then left to the system, wherein it is protected by error correction. This is shown in Figure 3.1.

The proposed in-field repair strategy is summarized as follows:

- Apply self-test and repair in field
- Reconfigure the address mapping

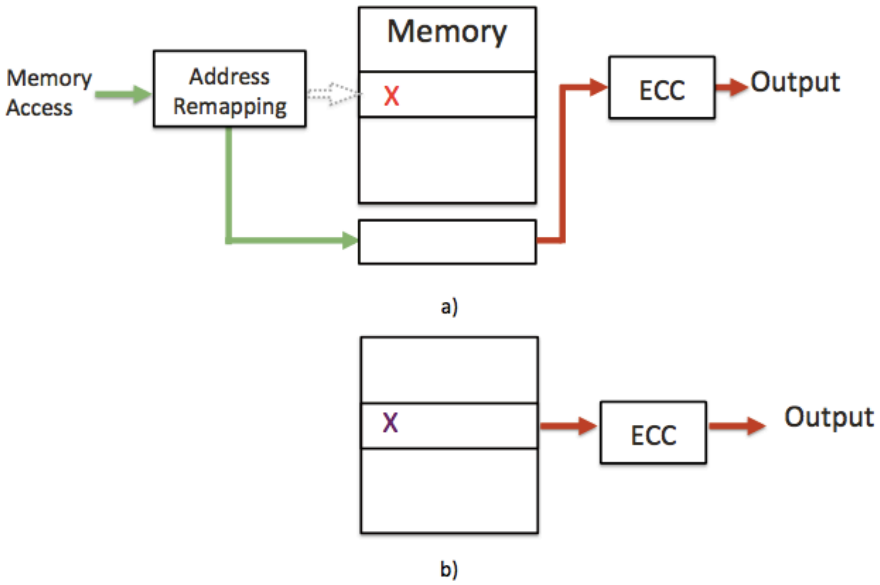

Figure 3.1: Proposed In-Field Strategy a) Repair word with 1 faulty bit if spare is available b) ECC can handle, if no more spares

- Repair a word if it has 2 or more faulty bits. Remapped word with 1 faulty bit is cancelled if no more spare available.
- Repair a word if it has 1 faulty bit and a spare is available. If no spare is available, do not repair the word and leave it to be corrected by ECC.

3.2 Memory Architecture Using Combination of In-Field Self-Repair and ECC

In this section, the specifics of the proposed memory architecture are described in detail. The proposed memory architecture is based on the work [5]. It is composed of ECC-based memory, memory BIST, diagnosis CAM and remap CAM as shown in Figure 3.2. We assume that an effective hardware repair has already been applied at production test, and memory used by the system includes enough number of spare words. Also, our proposed strategy can be applied to cases

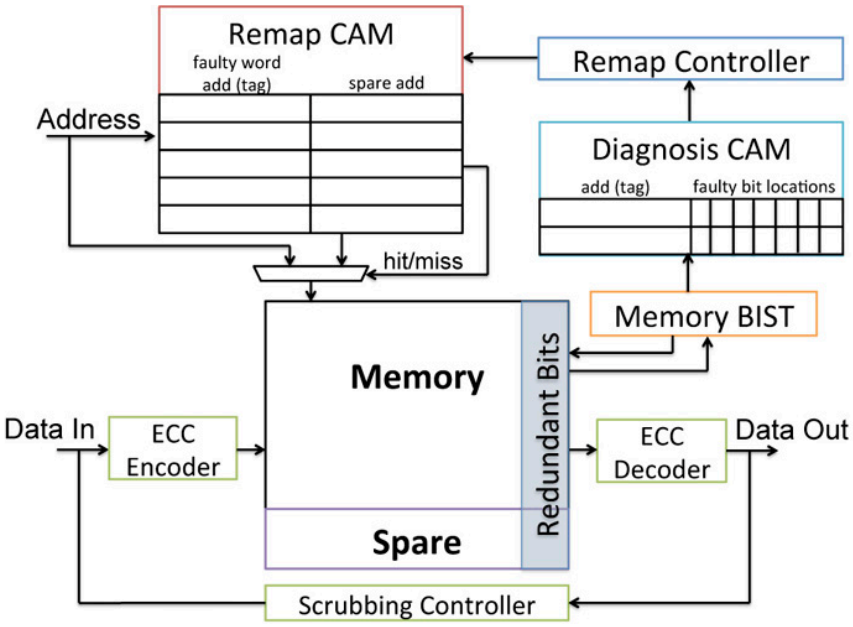

Figure 3.2: Proposed ECC-Based Memory Architecture

wherein correctable words are included in the shipped memory devices.

3.2.1 Architecture Details

The proposed architecture operates in two modes, 1) user mode, which is during normal operation, and 2) test mode, which is during test operation. The components in operation for each mode are described in detail.

User Mode

The user has access to the memory architecture. The ECC-based memory is composed of 1) user memory, which is available for normal read or write operations, and 2) spare memory, which can be reconfigured for repair purposes. Given a read operation for memory word, the data flow is as shown in Figure 3.3. Before accessing the user memory, the remap CAM is first searched. Data output is then accessed from the user memory if there is no hit, and from spare memory if there is a hit. The remap CAM, ECC encoder/decoder circuits and scrubbing controller are all operational. The parts are detailed as follows:

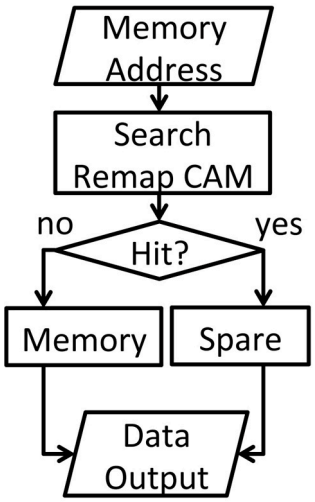

Figure 3.3: Data Flow for User Mode

1. Remap CAM: The remap CAM is active and a given original address is remapped to a spare address if the original address has been identified as a word to be repaired.

2. ECC Encoder/Decoder Circuits: ECC ensures correct output at any time a correctable error occurs.

3. Scrubbing Controller: Scrubbing controller periodically re-writes the data in its original location. Scrubbing eliminates soft errors if it is correctable through ECC. Given that a word with 1-bit soft error gets corrected via scrubbing, it becomes ready for an occurrence of another soft error.

Test Mode

Memory systems are physically organized into several blocks [19]. We assume that each block becomes idle periodically, and test can be applied when it is idle. We can apply periodic self-test and diagnosis separately on each memory block. The test mode operates as shown in Figure 3.4. BIST is firstly applied to find faulty words, then the found faulty word addresses and faulty bit locations are placed in the diagnosis CAM, and then based on the proposed strategy, the remap CAM is reconfigured based on operations listed in Table 3.1. The parts

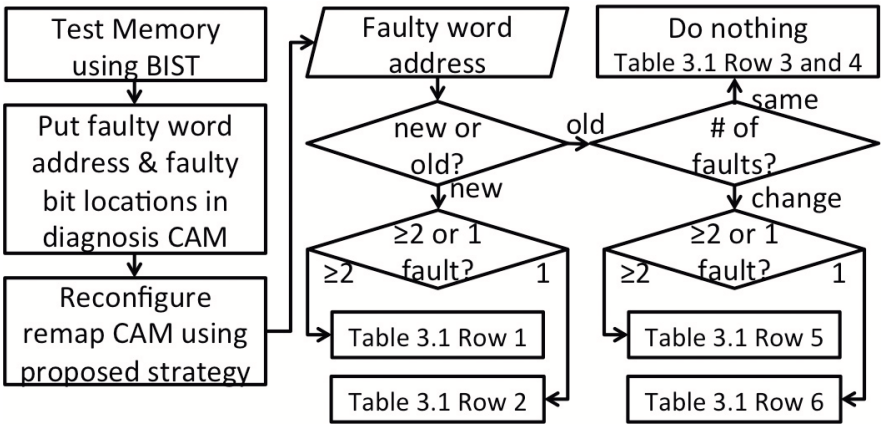

Figure 3.4: Data Flow for Test Mode

are detailed as follows:

1. BIST: BIST identifies faulty memory words. A March-like test repeatedly performs write and read operations in memory words. For each period of self-test, faulty bits in memory word are identified.

2. Diagnosis CAM: A diagnosis CAM is adapted [5] to classify faulty words based on the number of faulty bits. It stores the address of the faulty word, and the faulty bit locations within the word, once a faulty bit is detected during test. The address of the word is used as a tag to access the word in the diagnosis CAM. During test, though the same bit may be identified to be faulty several times, or multiple bits in the same words are identified to be faulty at different test timings, they are well integrated in the diagnosis CAM and every faulty word can be classified based on the number of faulty bits in a word.

3. Remap Controller: After the diagnosis CAM has stored all the faulty word addresses of the memory, this information is sent to a remapping controller. The remapping controller decides how the faulty word addresses are stored on the remap CAM depending on the number of faulty bit locations in the faulty word. It also updates the contents of the remap CAM on the succeeding self-test and repair periods.

3.2.2 Memory Elements

The role of the memory elements in the proposed architecture during the different modes of operation are described in detail as well.

During User Mode

- The memory is available to the user.
- The remap CAM is in effect.
- The spare memory is accessed when the called memory address is stored in the remap CAM.
- The diagnosis CAM is not operational during user mode.

During Test Mode

- The memory available to the user is subjected to the proposed in-field adaptive self-test and repair strategy.
- The spare words are included when test is performed to all words in memory. If a spare word is found to be faulty, a flag bit is used to set that the spare word is excluded from words that can be used for repair [4].
- Both CAMs may also be subjected to hard and soft errors. Similar to the the procedure adapted for spare words, if an entry of the CAM is found to be faulty, a flag bit in the tag field of the CAM is used to set that the tag field is excluded from words to be matched among all entries in the CAM [4]. The CAM is tested and repaired as described in [5].

3.3 Proposed Remap Controller Implementation

To realize the in-field repair strategy, a remap controller that executes an adaptive reconfiguration method is implemented. The remap CAM is reconfigured based upon the proposed strategy and the cases for remap CAM operation.

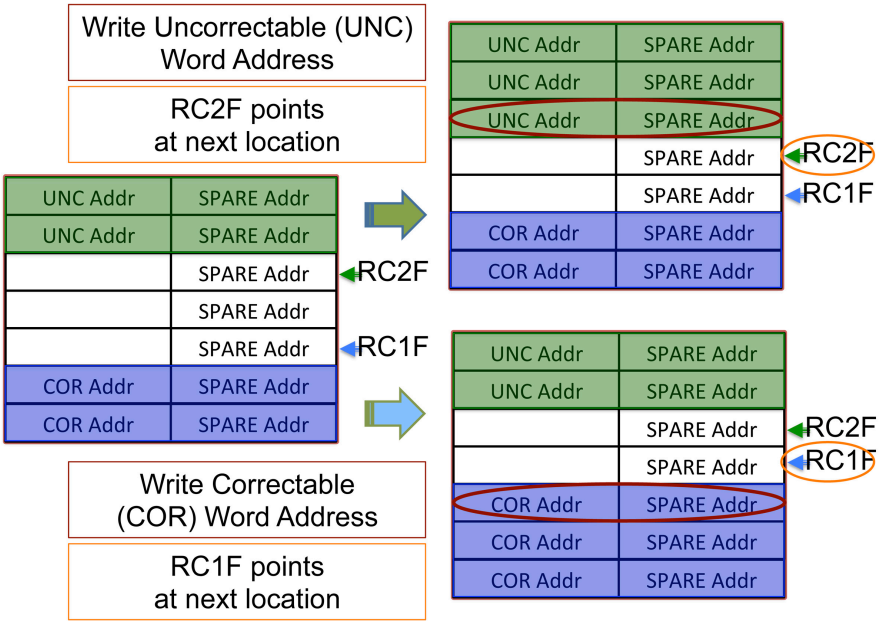

Figure 3.5: Writing New Entry when Remap CAM is Not Full

3.3.1 Reconfiguring the Remap CAM

The remap CAM is divided into two areas based on fault classification and we use two address counters RC2F and RC1F. The top part is used for uncorrectable words and the bottom part is used for correctable words. The counter RC2F points to the next address to be written into for uncorrectable words in the top part, and the counter RC1F points to the next address to be written into for correctable words in the bottom part. When there are enough unused spare words, all of the faulty words are repaired, wherein remap CAM is adequately maintained according to the in-field repair strategy, as explained in the previous section. When remap CAM is full, all of the spare words have been used for repair. After this, only uncorrectable words are added to the remap CAM. This is achieved by removing previously remapped correctable words.

Figures 3.5 and 3.6 show examples on how to reconfigure the remap CAM where we assume both RC2F and RC1F point to a valid spare memory word. Figure 3.5 shows a case where the remap CAM has enough unused spare words, and a new faulty word is written into the remap CAM. Since the word is a new entry, the address is not yet stored in the remap CAM. In a case where a new

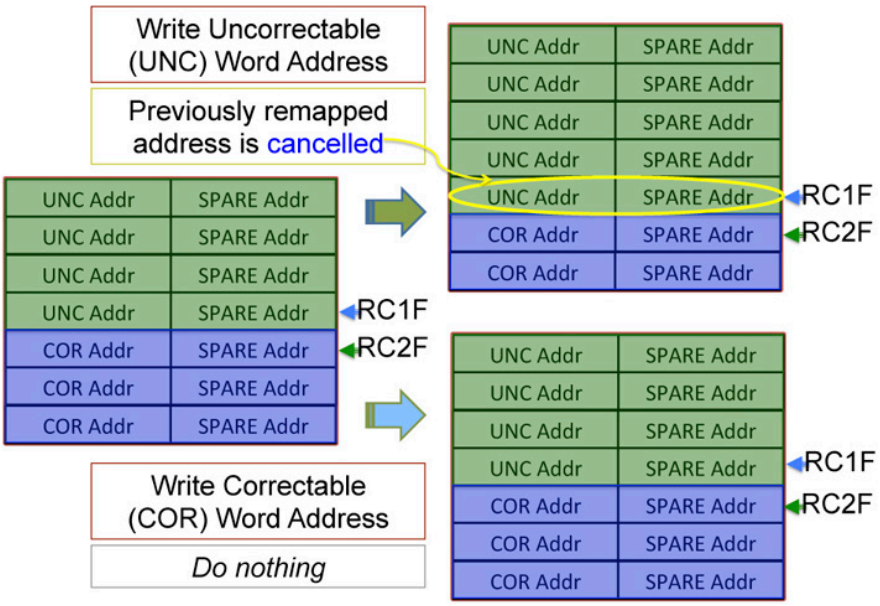

Figure 3.6: Writing New Entry when Remap CAM is Full

entry is an uncorrectable word, the address is written at the location pointed to by RC2F, and then RC2F is decremented. In a case of correctable word, its address is written at the location pointed to by RC1F, and then RC1F is incremented. Figure 3.6 shows a case where remap CAM has no more spare words and new faulty word is found. In a case where a new entry is an uncorrectable word, the address is written at the location pointed to by RC2F, and then RC2F is decremented. Since an uncorrectable word is given priority in our in-field repair strategy, the uncorrectable word address is overwritten into the entry which previously stored a correctable word address. On the other hand, if a new entry is a correctable word address, no action is done since priority is given to uncorrectable word addresses.

3.3.2 Remap CAM Operations

Table 3.1 shows all the possible cases for remap CAM operation. The left column shows 2 types of faulty word address entry. The remap controller classifies the faulty word address to be written into the remap CAM as either a new entry or an old entry. When a faulty word address is compared to the contents of the remap

Table 3.1: Remap CAM Cases

Faulty Word Address Entry	Number of Faulty Bits		Remap CAM Condition		
	Previous Test Period	Current Test Period	Enough	Full-1†	Full-2‡
new	-	≥2	1) write (RC2F) 2) decrement (RC2F)	1) write (RC2F) 2) decrement (RC2F) 3) decrement (RC1F)	fail
new	-	1	1) write (RC1F) 2) increment (RC1F)	do nothing	do nothing
old	≥2	≥2	do nothing	do nothing	do nothing
old	1	1	do nothing	do nothing	n/a
old	1	≥2	1) decrement (RC1F) 2) copy (hit_addr,RC2F) 3) copy (RC1F, hit_addr) 4) decrement (RC2F)	1) decrement (RC1F) 2) swap (hit_addr, RC1F) 3) decrement (RC2F)	n/a
old	≥2	1	1) increment (RC2F) 2) copy (hit_addr, RC1F) 3) copy (RC2F, hit_addr) 4) increment (RC1F)	1) increment (RC2F) 2) swap (hit_addr, RC2F) 3) increment (RC1F)	same as Full-1

Full-1†: both correctable and uncorrectable word addresses Full-2‡: uncorrectable word addresses only

CAM, and if it gives a miss, this means it is not yet stored at the remap CAM and the entry is new. If it gives a hit, it is denoted as a *hit_addr*, which means it is already stored at the remap CAM, and the entry is old. The middle column shows the number of faulty bits in a self-test period. Besides executing the in-field repair strategy for new entries, the remap controller considers whether the number of faulty bits varies or remains the same in a self-test period for old entries, and executes as such the corresponding operations. The operations *write*, *increment* and *decrement* take address pointers as parameters, where *write* means storing a faulty word address at the address in remap CAM pointed by the address pointer, while *increment* and *decrement* means to increment or decrement values of the address pointer. For the *copy* operation, the contents pointed by an address pointer (left operand) is copied to a destination address (right operand). For the *swap* operation, the contents pointed by the address pointers is swapped. The right column shows remap controller operations depending on the number of stored entries of the remap CAM. The remap controller executes operations whether a) the remap CAM has enough space to write a new entry, b) the remap CAM is full composed of both uncorrectable and correctable word addresses, or c) the remap CAM is full of uncorrectable word addresses. Memory fails when the remap CAM is full with uncorrectable words, and a new entry occurs.

3.4 Reliability Model Using Combination of In-Field Self-Repair and ECC

In this section, the stochastic model used in order to evaluate the reliability of the proposed strategy is explained. Also, the environment, under which this evaluation is performed, is described, specifically the periodicity of the time when self-test and repair, as well as memory scrubbing, is applied.

Reliability of the proposed strategy is evaluated under assumption that hard and soft errors occur with Poisson distributions. Let λ_h and λ_s be error rates of a single bit for hard and soft errors, respectively. Let N and N_s denote the number of words and spare words, respectively. The memory system provides N words to users among $N + N_s$ words.

Reliability of a memory is evaluated in the interval $[0, t]$. This is the probability

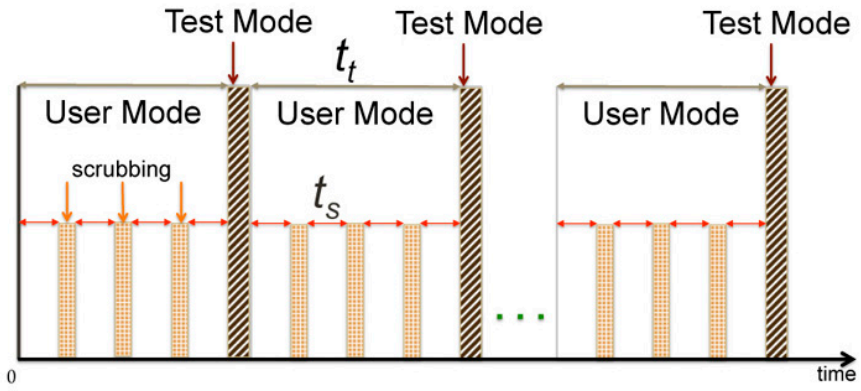

Figure 3.7: Self-test and repair and scrubbing

that a system is working well during $[0, t]$, which means the system has no uncorrectable word during $[0, t]$. Periodic self-test and repair and periodic scrubbing protect memory from hard and soft errors. Since scrubbing is more frequently applied than self-test, the interval $[0, t]$ is divided into several self-test periods and one self-test period is further divided into scrubbing periods as shown in Figure 3.7. Let t_t and t_s denote the lengths of self-test period (an interval between two successive self-tests) and scrubbing period (an interval between two successive scrubbings), respectively. The system is reliable if all the past and current self-test periods are reliable, and one self-test period is reliable if all the scrubbing periods in the self-test period are reliable. At any given moment t, a system works well, or has no uncorrectable word, if every user word has 1) no error, 2) one hard error and no soft error, or 3) no hard error and one soft error.

3.4.1 Probability of Words

Probability due to Hard Error

The Poisson distribution gives the probability that no hard error occurs for one word during $[0, t]$ as

$$P_{h0}(t) = e^{-\lambda_h nt} \tag{3.1}$$

where n is the number of bits in a codeword, the probability that $n-1$ bits have no hard error is given as

$$P_{h0}^{n-1}(t) = e^{-\lambda_h(n-1)t} \tag{3.2}$$

and the probability that exactly one bit gets hard error during period $[0,t]$ is given as

$$P_{h1}(t) = n\left(1 - e^{-\lambda_h t}\right) \cdot e^{-\lambda_h(n-1)t} \tag{3.3}$$

The probability that 2 or more bits get hard errors during period $[0,t]$ is given as

$$P_{>h1}(t) = 1 - P_{h0}(t) - P_{h1}(t) \tag{3.4}$$

and the probability that 1 or more bits get hard errors during period $[0,t]$ is given as

$$P_{\geq h1}(t) = 1 - P_{h0}(t) \tag{3.5}$$

Probability due to Soft Errors

The probability that no soft error occurs during period $[0,t]$ is given as

$$P_{s0}(t) = e^{-\lambda_s nt} \tag{3.6}$$

the probability that $n-1$ bits have no soft error is given as

$$P_{s0}^{n-1}(t) = e^{-\lambda_s(n-1)t} \tag{3.7}$$

and the probability that exactly one bit gets a soft error during period $[0,t]$ is given as

$$P_{s1}(t) = n\left(1 - e^{-\lambda_s t}\right) \cdot e^{-\lambda_s(n-1)t} \tag{3.8}$$

Probability due to Scrubbing

Assume that a word has no hard error during $[0,t]$. During this interval, the word works well if the word does not have multiple soft errors at the same time. Since scrubbing eliminates soft errors if each word has at most one soft error, the

probability that at most one bit has soft error at the same time during $[0, t]$ if there is no soft error at 0 is given as

$$P_{\leq s1}(t) = (P_{s0}(t_s) + P_{s1}(t_s))^c \cdot (P_{s0}(t_e) + P_{s1}(t_e)) \tag{3.9}$$

where $c = \lfloor t/t_s \rfloor$ is the number of scrubbings in $[0, t]$, and $t_e = t - ct_s$ is the elapsed time after the last scrubbing.

3.4.2 Reliability of Word for 1st Self-Test Period

The reliability for one word during $[0, t]$ within the $1st$ self-test period is given as

$$R_w^0(t) = P_{h0}(t) \cdot P_{\leq s1}(t) + P_{h1}(t) \cdot P_{s0}^{n-1}(t) \tag{3.10}$$

where any word has no error at time 0 is assumed.

3.4.3 Reliability of Memory using Traditional Strategy

Using the reliability of word for the $1st$ self-test period, the reliability of memory system during $[0, t]$ within the $1st$ self-test period is given as

$$R^0(t) = \left(R_w^0(t)\right)^N \tag{3.11}$$

This expresses as well the reliability of a memory system that does not repair faulty words with one hard error.

3.4.4 Reliability of Memory for 1st Self-Test Period

The reliability of memory during the $1st$ period is given as

$$R_{period}(1) = R^0(t_t) \tag{3.12}$$

3.4.5 Reliability of Word during One Test Period

The reliability of one word during one test period for an interval $[t, t + t_e]$, where t is the time at the beginning of the current self-test period and t_e is the elapsed time in the current test period, is given as

$$R_w^0(t_e) = P_{h0}(t_e) \cdot P_{\leq s1}(t_e) + P_{h1}(t_e) \cdot P_{s0}^{n-1}(t_e) \tag{3.13}$$

if there is no hard error at t, and

$$R_w^1(t_e) = P_{h0}^{n-1}(t_e) \cdot P_{s0}^{n-1}(t_e) \tag{3.14}$$

if there is 1 hard error at t.

3.4.6 Reliability of Self-Test Period

The memory system is still reliable if one of the following two conditions hold as a result of self-test: 1) there are at least N words with 0 hard error, or 2) there are less than N words with 0 hard errors, but at the same time, there at least N words with 0 or 1 hard errors.

The reliability for the k-th period is given as

$$R_{period}(k) = \frac{P_0(k)}{P_0(k) + P_{0+1}(k)} \left(R_w^0(t_t)\right)^N + \sum_{i=0}^{N-1} \frac{P_{0(i)+1}(k)}{P_0(k) + P_{0+1}(k)} \left(R_w^0(t_t)\right)^i \left(R_w^1(t_t)\right)^{N-i} \tag{3.15}$$

For the condition that there are at least N words with 0 hard error in the memory at the beginning of the k-th period, the probability is given as

$$P_0(k) = \sum_{i=N}^{N+N_s} \binom{N+N_s}{i} \left(P_{h0}((k-1)\,t_t)\right)^i \cdot \left(P_{\geq h1}((k-1)\,t_t)\right)^{N+N_s-i} \tag{3.16}$$

where there are i words with 0 hard errors, and $N + N_s - i$ words with 1 or more errors.

For the condition that there are i words ($i < N$) with 0 hard error and at least N words has at most one hard error at the beginning of the k-th period can be expressed as

$$P_{0(i)+1}(k) = \sum_{j=N-i}^{N+N_s-i} \binom{N+N_s}{i+j} \binom{i+j}{j} \left(P_{h0}((k-1)\,t_t)\right)^i \cdot \left(P_{h1}((k-1)\,t_t)\right)^j \left(P_{>h1}((k-1)\,t_t)\right)^{N+N_s-i-j} \tag{3.17}$$

where there are i words with 0 hard errors, j words with 1 hard errors, and $N + N_s - i - j$ words with 2 or more errors.

Then, for the condition that there are less than N words with 0 hard errors and there are at least N words with 0 or 1 hard errors at the beginning of the k-th period, the probability is given as

$$P_{0+1}(k) = \sum_{i=0}^{N-1} P_{0(i)+1}(k) \tag{3.18}$$

3.4.7 Reliability of Memory System using Proposed Strategy

To complete the expression for the reliability as a function of time, the reliabilities for the all the previous $(k-1)$-th periods are multiplied to the current period until the elapsed time, and the reliability of memory is given as

$$R(t) = \prod_{p=1}^{k-1} R_{period}(p) \times [\ \frac{P_0(k)}{P_0(k)+P_{0+1}(k)} \left(R_w^0(t_e)\right)^N + \sum_{i=0}^{N-1} \frac{P_{0(i)+1}(k)}{P_0(k)+P_{0+1}(k)} \left(R_w^0(t_e)\right)^i \left(R_w^1(t_e)\right)^{N-i}\ ] \tag{3.19}$$

where t is in the k-th period, and $t_e = t \bmod t_t$ is the time elapsed after the last self-test.

3.5 Experimental Results

In this section, the evaluation results of the reliability model, and hardware overhead of the remap controller are presented.

3.5.1 Reliability Evaluation

The reliability is evaluated using a data analytical and graphing software. Nominal values are chosen as baseline as shown in Table 3.2. The number of bits in a code word n considers 16-bit memories and 5 redundant bits to provide error correction. Soft errors must occur much more frequently that hard errors, wherein $\lambda_h \ll \lambda_s$, since soft errors are transient phenomenon, while hard errors are physical defects that occur after continued usage of the device. The scrubbing interval t_s is set to occur every 6 minutes, while self-test and repair is set to occur every

Table 3.2: Baseline Memory Parameters

Parameters	Definition	Values
N	number of words	1e+5
N_s	number of spare words	50
n	number of bits per word	21
λ_h	hard error rate (errors per hour)	10^{-11}
λ_s	soft error rate (errors per hour)	10^{-7}
t_t	time between self-test and repair (hours)	240
t_s	time between scrubbing (hours)	0.1

10 days. The reliability of the proposed strategy is compared to the reliability of the traditional combined approach strategy [2,5], where only uncorrectable words are repaired, while correctable words are left to the ECC.

From the experts' hearings, it is known that a part of the power plant products requires high reliability for lifetime longer than 20-50 years, though usual products and applications require reliability of 10 years for normal systems, and 20 years for automotive. Throughout 50 years, including the 10-year and 20-year marks, the results show that the proposed strategy has better reliability than the traditional strategy.

The parameters are varied in order to observe how the proposed strategy improves reliability. Figure 3.8 shows the results of the reliability evaluation for various memory sizes. The reliability of the proposed strategy is close to 1 all throughout the observed range, except for the case of 512K words. Also, the proposed strategy is more reliable by several decades than the traditional strategy.

Figure 3.9 shows the results for various spare word sizes. All the plots for the traditional strategy coincide with each other. This shows that even if there are enough spare words, if correctable words are not repaired, the traditional strategy is not able to mitigate the errors, and system reliability is severely degraded. The plots of both 50 and 500 spare words for the proposed strategy coincide with each other. This shows that for a given baseline of error rate and for a given time period that the reliability of the system is observed, a minimum number of spare words

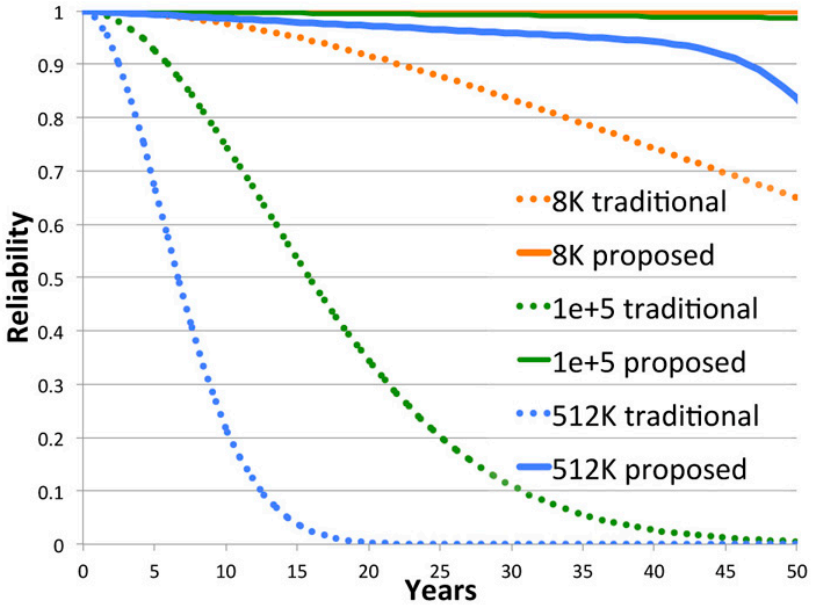

Figure 3.8: Reliability of proposed strategy vs traditional strategy for various memory sizes

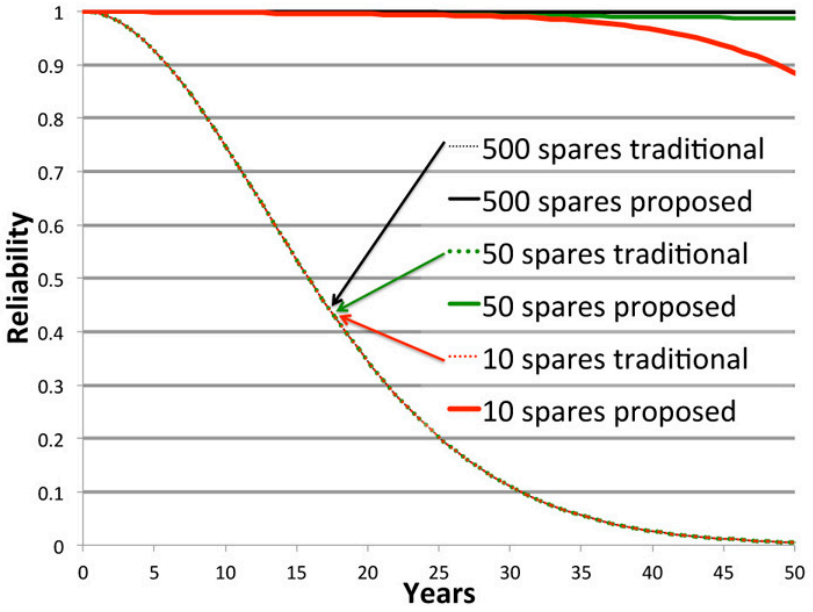

Figure 3.9: Reliability of proposed strategy vs traditional strategy for various spare word sizes

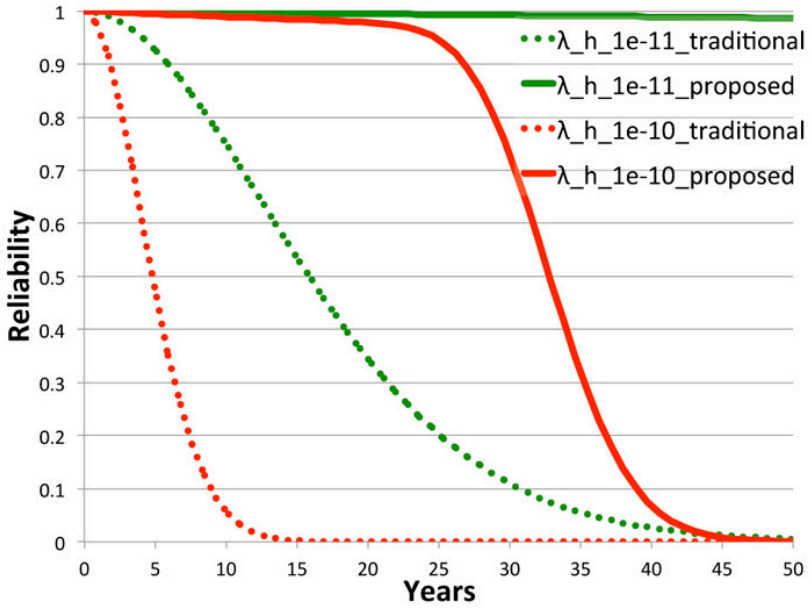

Figure 3.10: Reliability of proposed strategy vs traditional strategy for varying hard error rate given soft error rate λ_s=10^{-7}

can be enough to maintain reliability.

Figure 3.10 shows that case when hard errors occur more often. Though the proposed strategy may use up all of the spare words within several decades, it has better reliability than the traditional strategy. Figure 3.11 shows the case when soft errors occur more often. The proposed strategy provides higher reliability than the traditional strategy, though the system reliability degrades quickly. For high soft error rates, the system might require stronger soft error protection.

Figure 3.12 and Figure 3.13 show the results for various word lengths, where two numbers in the legend mean lengths of an original word and redundant bits for ECC. Memories with longer word lengths are more susceptible to errors, especially for codewords of 522 bits, which may require stronger correction schemes. For all the word lengths, the proposed strategy provides greater reliability than traditional stately.

Figure 3.14 shows the results for varying self-test and repair periods. The result shows that as long as the proposed strategy performs self-test and repair on the ample period needed, it will be able to protect the memory system. When

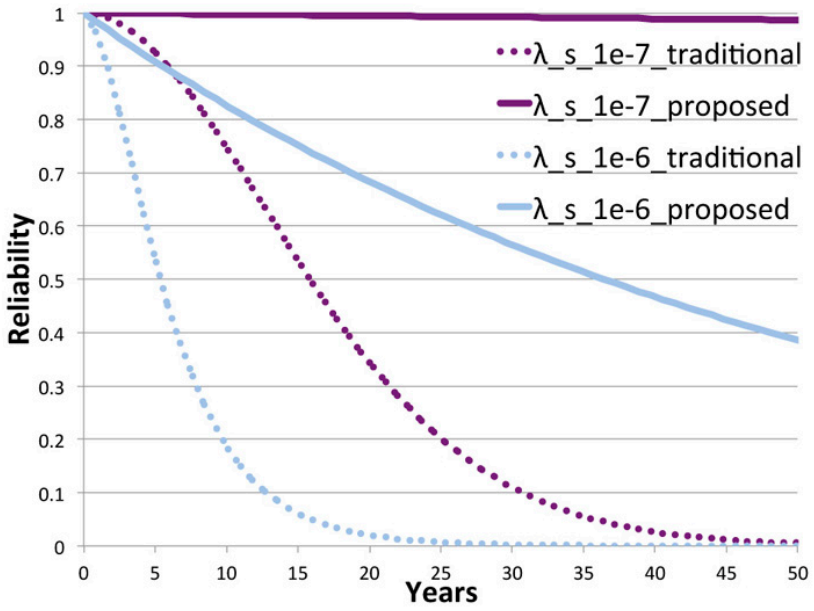

Figure 3.11: Reliability of proposed strategy vs traditional strategy for varying soft error rate given hard error rate λ_h=10^{-11}

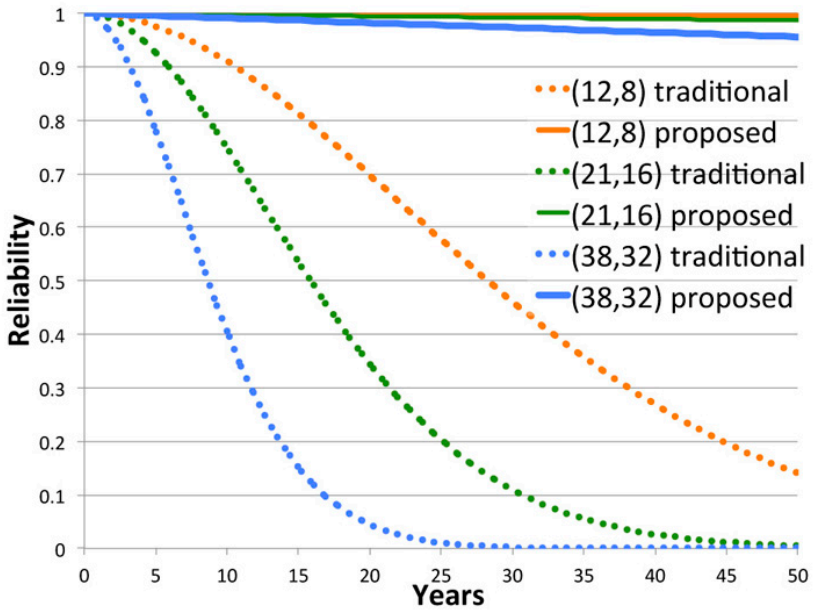

Figure 3.12: Reliability of proposed strategy vs traditional strategy for various word lengths up to 32 bits

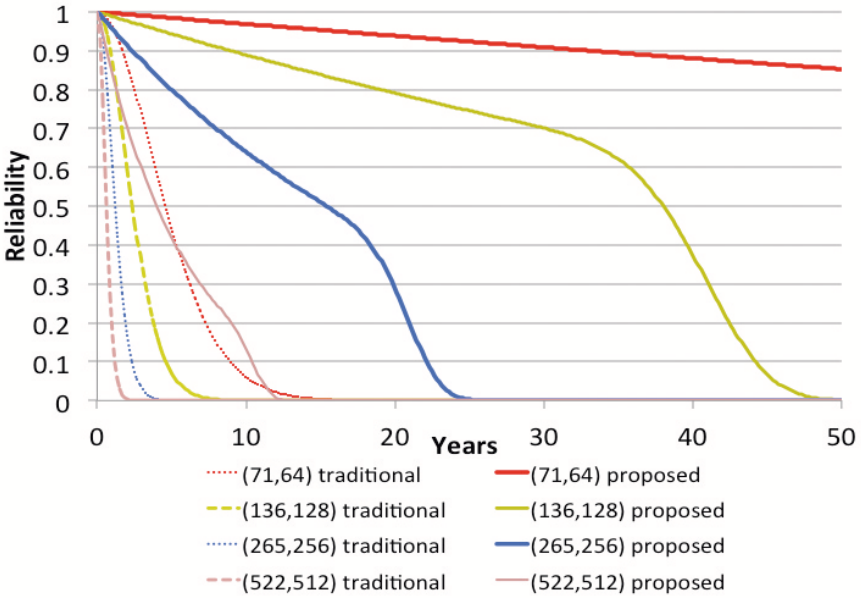

Figure 3.13: Reliability of proposed strategy vs traditional strategy for various word lengths up to 512 bits

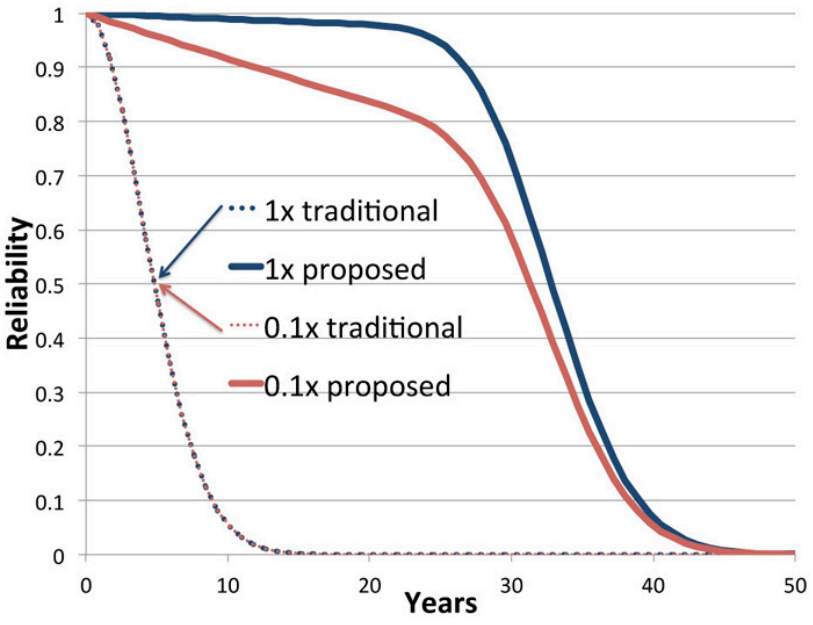

Figure 3.14: Reliability of proposed strategy vs traditional strategy for varying self-test and repair periods

Table 3.3: Memory Elements Used for Proposed Architecture

Memory Element	Number of Words	Word Size (in Bits)
Memory	1e+5	21
Spare	50	21
Remap CAM	50	34
Diagnosis CAM	256	38

self-test and repair is done in a slower rate, the reliability of the system is affected.

All in all, the proposed strategy shows enhanced reliability performance compared to the traditional strategy.

3.5.2 Hardware Overhead

The remap controller of the proposed in-field repair strategy using the Synopsys SAED90nm_EDK library is implemented. Controllers were implemented for 5 types of memory sizes given a fixed number of words for spare memory, diagnosis and remap CAM. The size of each entry for the diagnosis and remap CAM depends on memory size. The overhead introduced by the remap controller to determine the area cost is evaluated. Also, the power and latency overhead of diagnosis CAM and remap CAM is evaluated, and the method and tool used to perform this evaluation is presented. Then, the power and latency overhead of the periodic test strategy by assessing the number of operations needed to perform test are evaluated. A method that minimizes power and latency is cited as well.

Area Overhead

Table 3.3 shows the values of memory elements for a case of 1e+5 words. The remap CAM has the same number of entries as the spare words, but has twice the address width, since each entry consists of a faulty word address and the corresponding spare word address. The diagnosis CAM has 256 word entries with word size of 38 bits, 17 bits for the faulty word address and 21 bits, which is the memory word size, since these are where the faulty bit locations are stored.

Table 3.4: Hardware Overhead of Remap Controller

Number of Words	Address Bit Count	Gate Count
8K + 50 spare	14	1467
64K + 50 spare	17	1570
1e+5 + 50 spare	17	1599
256K + 50 spare	19	1663
512K + 50 spare	20	1711

*K means exactly 1,024 bits

The rest of the memory sizes use an extra addressing bit to be able to map into the spare addresses. For example, a memory with 8K words is addressed with 13 bits + 1 more bit to access the spare words, so in total, 14 bits are used. Table 3.4 shows the experimental results gathered for increasing memory sizes, where areas are evaluated in 2-input NAND gate-equivalent. The size of the remap controller slightly increases as memory words increase. This is because the number of addressing bits logarithmically increases with increase in memory size. For all the types, the hardware overhead is less than 1,800 gates, which is small compared to the total number of gates for an SoC. It also shows that the remap controller is scalable.

Power and Latency Overhead

The power and latency overhead using memory elements shown in Table 3.3 are evaluated. The power overhead is mainly contributed by the CAM operation due to the comparison circuitry activated in parallel. Both CAMs are relatively small in size. The bigger diagnosis CAM is only used during test mode, while the smaller remap CAM is used during test mode, and during user mode, where remapping is in effect. The CAM size limits the number of operations performed by the CAM. We use CACTI [20] to evaluate power and latency for 90nm technology. CACTI is used for cache memories, and does not fully support CAM evaluation. Nevertheless, the tool can be used by carefully choosing parameters that are close to that of the CAMs in Table 3.3. CACTI can evaluate set associative cache memories with at least 16 sets, associativity of a power of 2, and block

Table 3.5: CAM Power Overhead and Access Time

Parameter	Diagnosis CAM	Remap CAM
Dynamic Power (W)	0.0995	0.1066
Leakage Power (W)	0.0309	0.0022
Total Power (W)	0.1304	0.1088
Access Time (ns)	5.94312	1.8076

size of at least 8 bytes. On the other hand, the CAMs in Table 3.3 require full associative (single set) caches with 256 entries (associativity) and 17-bit tag and 21-bit data (block size) for the diagnosis CAM, and 50 entries and 17-bit tag and 17-bit data for the remap CAM. Caches with 16 sets and 8-byte (64-bit) block, and 256 entires for the diagnosis CAM and 64 entries for the remap CAM are evaluated. In this setup, each of the diagnosis CAM and the remap CAM is considered as one of the sets of a set associative cache. However, when accessing a cache, only one set becomes active, and evaluation of power and latency of the CAMs with a slight overestimation is considered. The results are shown in Table 3.5.The dynamic power and leakage power for each set of the CAMs is found to be relatively small. Also, the access time, and thus latency overhead, is found to be relatively small.

The power and latency overhead of periodic test strategy are evaluated. The memory BIST, which uses a March-like self-test for a word-based memory organization, is assessed. The MATS+ [21] is used to perform self-test and has test time of $5 * n/B$, where n is number of cells and B is number of bits in a word. Given that memory organization is composed of N user words and N_s spare words, the total time to perform memory test is $5 * (N + N_s)$ operations, which are relatively few.

Also, the number of times the CAMs may be accessed during periodic test and repair, which is dependent on the number of faults found, is assessed. Given the memory in Table 3.3, for a hard error rate of $\lambda_h = 10^{-11}$, the faults that have occurred are expected to be $\sim$1.83 at the 10-year mark, and $\sim$3.64 at the 20-year mark, while for a hard error rate of $\lambda_h = 10^{-10}$, the faults that have occurred are expected to be $\sim$18.35 at the 10-year mark, and $\sim$36.4 at the 20-year mark.

From this, it is known that at first, less faults occur, and the number of accesses to the CAMs are infrequent. Then after some time, more faults occur, and the number of accesses to the CAMs increase. This implies that power or latency overhead to access CAMs during test mode is negligible in early life, and it is slightly increased with time. These overheads can be reduced by dividing a whole memory into blocks and applying the test and repair to each memory block [5].

3.6 Chapter Summary

Given an ECC-based memory architecture, a novel in-field repair strategy that repairs uncorrectable words and possibly correctable words is proposed. An adaptive reconfiguring method is executed by the implemented remap controller to realize the proposed in-field repair strategy. A stochastic reliability model of the proposed strategy is established based on Poisson distributions for hard and soft errors. The proposed strategy enhances memory reliability in field compared to the traditional strategy, which repairs uncorrectable words only. Also, the hardware overhead of area, power and latency is found to be small. These show that the proposed strategy extends memory lifetime in practical use.

4 Reliability Enhancement Using Combination of Aging-aware Adaptive In-Field Self-Repair and ECC

In this chapter, we propose to enhance reliability further by introducing a new reliable memory architecture that combines aging test and the ECC-based memory architecture with in-field self-repair proposed from the previous chapter and uses a novel aging-aware in-field repair strategy. First, we propose a 3-state model of a memory cell where agedness is considered as a state of a memory cell. We also derive the probabilities when the state of a memory cell included in a memory word is considered as aged for a memory word organization. We describe in detail a reliable aging-aware memory architecture, and the further enhancement in reliability is evaluated.

4.1 Aging and Memory Cells and Memory Words

Given that a memory cell is aged compared to an average memory cell, we now consider the three possible states of a memory cell: *healthy, aged, and faulty.* Both healthy and aged cells are considered non-faulty, but an aged cell is more susceptible to turn faulty. Degradation of SNM is one cause of aged cell. In this section, we introduce a 3-state model for memory cell states, then introduce five status types of memory words.

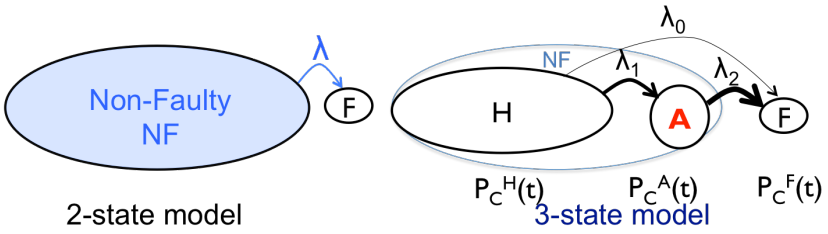

Figure 4.1: 2-State Model vs. 3-State Model for Memory Cell

4.1.1 State Model of Memory Cells

Figure 4.1 shows a conventional 2-state model and the proposed 3-state model for memory cells. The 2-state model has non-faulty and faulty states, and it is assumed that a non-faulty cell becomes faulty according to a Poisson distribution with a constant rate of λ. Note that such a constant rate can be assumed in the stable or useful life period of memory lifetime, and the analyses focus on this period.

In the three state model, a non-faulty state is divided into healthy and aged states. States of aged and faulty can be determined with SNM for example, and therefore, we can also assume that the rate that a healthy cell becomes aged or faulty is also constant. In the proposed 3-state model, we denote transition rates from healthy to faulty, healthy to aged and aged to faulty as constants λ_0, λ_1, and λ_2, respectively.

4.1.2 Probability of Memory Cell States

Let $P_C^H(t)$, $P_C^A(t)$, $P_C^F(t)$ denote probabilities that a cell is healthy, aged and faulty at time t, respectively. As shown in Figure 4.1, from 2-state and 3-state models, we derive the following equations:

$$\lambda_0 \cdot P_C^H(t) + \lambda_2 \cdot P_C^A(t) = \lambda \left(P_C^H(t) + P_C^A(t) \right) \tag{4.1}$$

$$\frac{d}{dt} P_C^H(t) = -\left(\lambda_1 + \lambda_0\right) P_C^H(t) \tag{4.2}$$

$$\frac{d}{dt} P_C^A(t) = \lambda_1 P_C^H(t) - \lambda_2 P_C^A(t) \tag{4.3}$$

$$P_C^H(t) + P_C^A(t) + P_C^F(t) = 1 \tag{4.4}$$

From Equation 4.1, we first obtain

$$\frac{P_C^H(t)}{P_C^A(t)} = -\frac{\lambda - \lambda_2}{\lambda - \lambda_0} \tag{4.5}$$

which shows that the ratio of $P_C^H(t)$ and $P_C^A(t)$ is a constant independent of time.

Let $P_C^H(0) = \alpha$, $P_C^A(0) = \beta$ and $P_C^F(0) = 0$, which implies that $\alpha + \beta = 1$, and $\frac{P_C^H(t)}{P_C^A(t)} = \frac{\alpha}{\beta}$. Then, we can derive the probabilities on cell states using Laplace transform:

$$P_C^H(t) = \alpha e^{-(\lambda_1+\lambda_0)t} \tag{4.6}$$

$$P_C^A(t) = \beta e^{-(\lambda_1+\lambda_0)t} \tag{4.7}$$

$$P_C^F(t) = 1 - P_C^H(t) - P_C^A(t) \tag{4.8}$$

Values of α and β can be determined as, for example, SNM of an aged cell is less than -3σ or -4σ with assumption of Gaussian distribution of SNM. Assume that λ_0 is so small that we can approximate $\alpha = 99.87\%$, $\beta = 0.13\%$, $\frac{\alpha}{\beta} = 768$, $\lambda_2 \approx 768\lambda$, $\lambda_0 \approx \frac{\lambda}{768}$, $\lambda_1 \approx \lambda$ for -3σ, and $\alpha = 99.997\%$, $\beta = 0.003\%$, $\frac{\alpha}{\beta} = 33332$, $\lambda_2 \approx 33332\lambda$, $\lambda_0 \approx \frac{\lambda}{33332}$, $\lambda_1 \approx \lambda$ for -4σ.

4.1.3 Word Status Types

Based on cell states, we introduce 5 word status types as shown in Figure 4.2. They are classified as: 1) $2F$: a word with 2 or more faulty cells, 2) $1FA$: a word with 1 faulty cell and 1 or more aged cells, 3) $1F0$: a word with 1 faulty cell and no aged cells, 4) A: a word with 1 or more aged cells, and lastly, 5) H: a word with all healthy cells. A $2F$ word is uncorrectable through the ECC as single error correction is employed. Words are vulnerable, or prone to be uncorrectable, in the order of $2F$, $1FA$, $1F0$, A and H. The priority for repair uses the same order as well, where $2F$ words have the most priority.

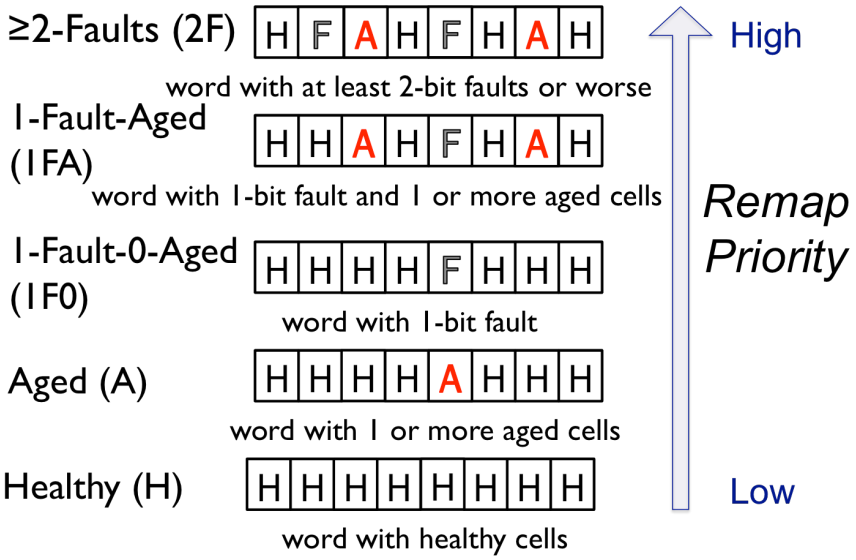

Figure 4.2: Word Status Types Based on Priority For Repair

4.1.4 Probability of Memory Word Status Types

Let $P_W^H(t)$, $P_W^A(t)$, $P_W^{1F0}(t)$, $P_W^{1FA}(t)$, $P_W^{2F}(t)$ denote probabilities that a word status is H, A, $1F0$, $1FA$, $2F$ at time t, respectively. These probabilities can be derived as follows:

$$P_W^H(t) = \left(P_C^H(t)\right)^n \tag{4.9}$$

$$P_W^A(t) = \sum_{i=1}^{n} P_W^A(t,i) = \sum_{i=1}^{n} \binom{n}{1} \left(\frac{\beta}{\alpha}\right)^i \left(P_C^H(t)\right)^n$$

$$= \left\{\left(1+\frac{\beta}{\alpha}\right)^n - 1\right\} \left(P_C^H(t)\right)^n \tag{4.10}$$

$$P_W^{1F0}(t) = \binom{n}{1} \left(P_C^F(t)\right) \left(P_C^H(t)\right)^{n-1} \tag{4.11}$$

$$P_W^{1FA}(t) = \sum_{i=1}^{n-1} P_W^{1FA}(t,i) = \sum_{i=1}^{n-1} \binom{n}{1} \left(P_C^F(t)\right) \binom{n-1}{1} \left(\tfrac{\beta}{\alpha}\right)^i \left(P_C^H(t)\right)^{n-1}$$

$$= \binom{n}{1} \left(P_C^F(t)\right) \left\{ \left(1 + \frac{\beta}{\alpha}\right)^{n-1} - 1 \right\} \left(P_C^H(t)\right)^{n-1} \tag{4.12}$$

$$P_W^{2F}(t) = 1 - P_W^H(t) - P_W^A(t) - P_W^{1F0}(t) - P_W^{1FA}(t) \tag{4.13}$$

Given the probabilities and the order of vulnerability due to status type of the memory words, a strategy must be devised in such a way that the most vulnerable word is given priority to be repaired. The architecture and the strategy needed to fulfill this strategy are explained in the succeeding sections.

4.2 Memory Architecture Using Combination of Aging-Aware Adaptive In-Field Self-Repair and ECC

In this section, a reliable memory architecture that combines ECC and in-field aging test and repair is described in detail. It is an enhancement from the architecture introduced in the previous chapter by using an extended in-field repair strategy that takes into account aging of memory cells.

4.2.1 Extended In-Field Repair Strategy

In the extended proposed strategy, the word status types are repaired, as shown in Figure 4.3. The more the vulnerable words, the higher priority for repair. It is summarized as follows:

- Apply self-test and repair in field
- For each detected faulty or aged word, reconfigure the remap CAM as follows:
  - Repair a word through remapping if spare space is not fully occupied with words with the same or higher repair priority. If there is no free spare word, cancel one word with lowest priority if it exists, and replace it with current word to be repaired

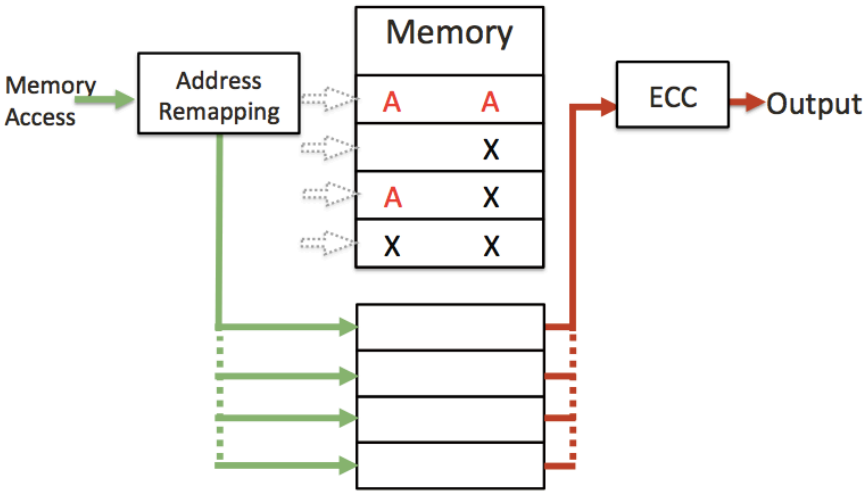

Figure 4.3: Extended In-Field Repair Strategy

  - Do nothing if spare space is occupied with words with the same or higher priority.

- Correctable or aged words may be used in field, with the output corrected by ECC if necessary.

4.2.2 Architecture Details

The main addition to the aging-aware architecture as compared to the previous architecture is the aging test. The aging-aware architecture is shown in Figure 4.4 operates in user mode and test mode. In user mode, a remap CAM isolates already identified faulty or aged words, and ECC and scrubbing protect users from memory errors. In test mode, memory BIST and aging test are applied, and the remap CAM is reconfigured. The various functionalities and the main additions are briefly reviewed below.

1. Remap CAM: A called address is remapped to a spare address if it is stored in remap CAM. It stores the memory words of various status types.

2. ECC Encoder/Decoder Circuits: They ensure correct output if the words can be corrected.

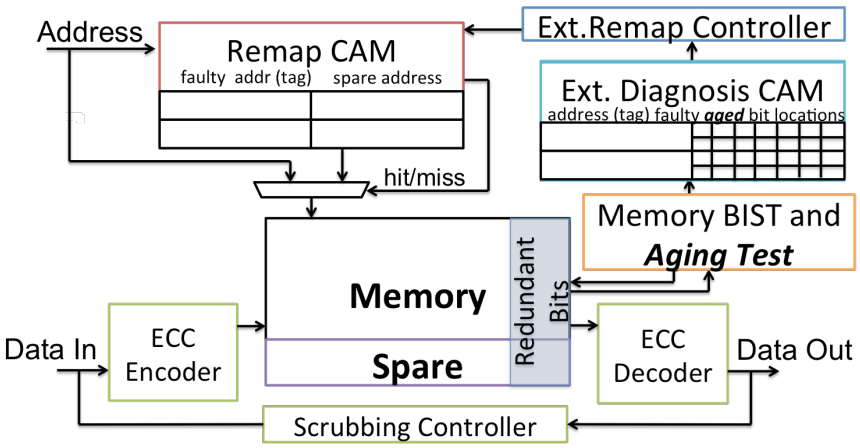

Figure 4.4: Proposed ECC-Based Aging-Aware Memory Architecture

3. Scrubbing Controller: It re-writes content of memory word through ECC and eliminates correctable soft errors within the word.

4. Memory BIST and Aging Test: Memory BIST identifies faulty memory cells, and aging test identifies aged cells.

5. Extended Diagnosis CAM: It diagnoses words with faulty or aged cells as either $2F$, $1FA$, $1F0$ or A words. It stores the addresses of the diagnosed words, and the bit locations of faulty and aged cells within the word.

6. Extended Remap Controller: It reconfigures the remap CAM based on vulnerabilities.

4.2.3 Extended Remap Controller

The remap CAM stores the original and remapped addresses of $2F$, $1FA$, $1F0$ and A words to be remapped as shown in Figure 4.5. From the top of the remap CAM, $2F$ words use the space with the space for $1FA$ words following, while $1F0$ words use the space starting from the bottom with the space for A words following upwards. This arrangement is employed to minimize rewrite traffic and retaining word status type contiguity. Four address counters, $RC2F$, $RC1FA$, $RC1F0$ and RCA, are used to mark word status type boundaries. Figure 4.6

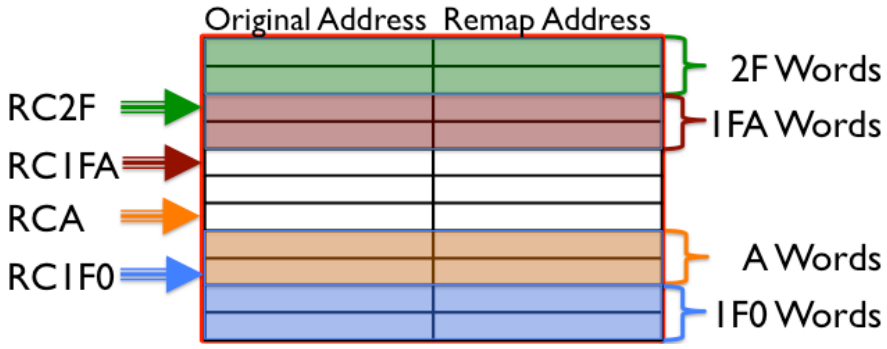

Figure 4.5: Remap CAM Using Extended In-Field Repair Strategy

shows an example where a new $2F$ word is inserted into the remap CAM. The space for $2F$ words is extended and the top $1FA$ word is moved to the bottom of space for $1FA$ words. It shows a similar operation for a new $1F0$ word, but from the opposite direction. Figure 4.7 shows a case where a new 2F word is inserted to a full remap CAM. In this case, spaces for $2F$ and $1FA$ are reconfigured similarly as Figure 4.6 and the top entry of the A word is overwritten. It can be seen in this case that the dropped A word is no longer in the remap CAM as priority has been given to the more vulnerable word. Concurrently, writing a new A word is forfeited since it is less vulnerable, and thus, with lower priority.

4.3 Reliability Model Using Combination of Aging-aware Adaptive In-Field Self-Repair and ECC

The reliability of the extended proposed strategy is evaluated under the condition where BIST and aging test is periodically applied together and scrubbing is also applied periodically.

Let N and N_s be the number of user words and spare words, respectively. The memory system utilizes N words among $N + N_s$ words. Memory system reliability is the probability that the system is working well in the interval $[0, t]$. This means the system has no unremapped $2F$ words during $[0, t]$. The interval $[0, t]$ is divided into several self-test periods and one self-test period is further

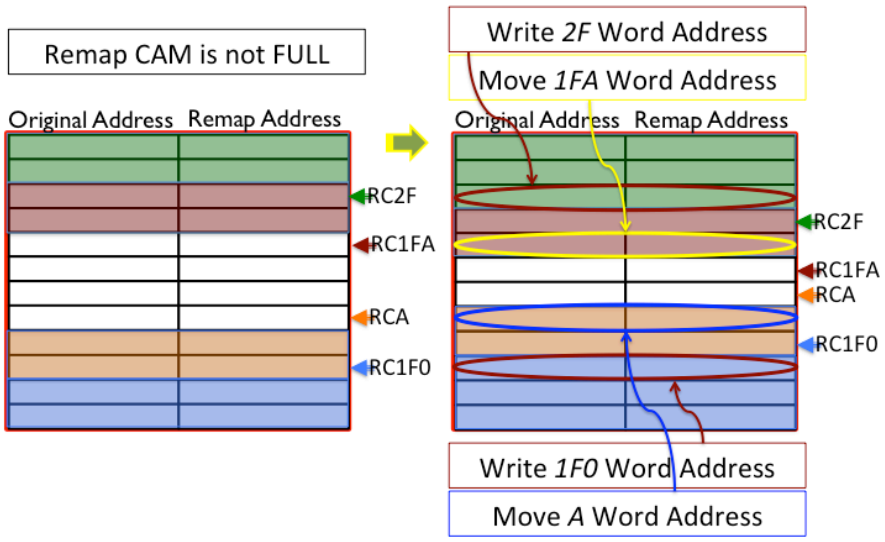

Figure 4.6: Writing New Entry when Extended Remap CAM is Not Full

divided into scrubbing periods. Self-test period is the time between consecutive self-tests, and scrubbing period is the time between consecutive scrubbings. Let t_t and t_s denote the lengths of self-test period and scrubbing period, respectively. The system is reliable if all the past and current self-test periods are reliable, and one self-test period is reliable if all the scrubbing periods in the self-test period are reliable.

This section is organized as follows:

1. The probabilities that occur at the cell level (bit level) are described first. The state probabilities describe the probability that a cell is healthy, aged or faulty, and they are the basis of the aging-aware reliability evaluation. They are described at the beginning of this chapter. The transition probabilities takes into account that the states of the cell may change after an interval. The cell probabilities also take into account that they are affected by soft error.

2. The probabilities at the word level are described based on the expressions formulated at the cell level. A word is composed of n bits.

3. The reliability of a memory word is expressed based on the word level

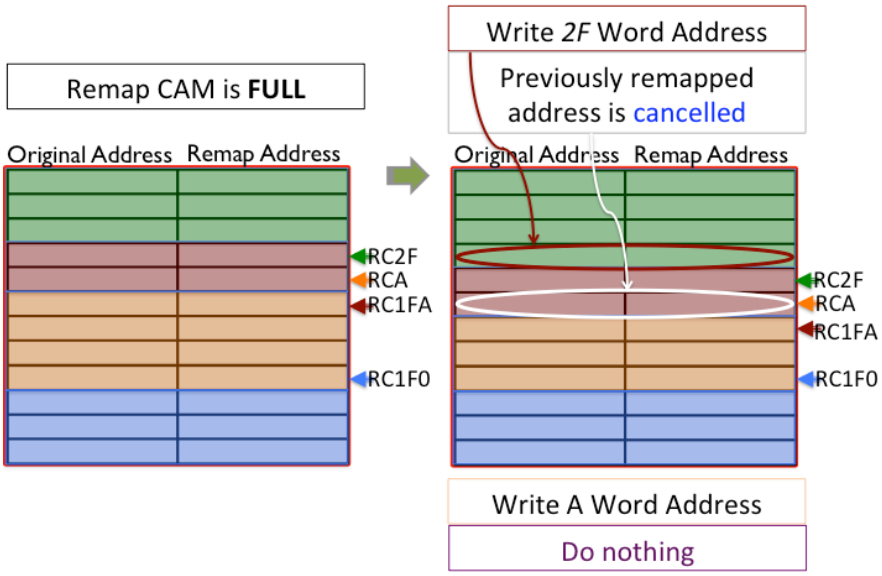

Figure 4.7: Writing New Entry when Extended Remap CAM is Not Full

probabilities such that it has at most 1 error at any given time T. These are the reliability expressions for the word status types.

4. The reliability of one self-test period is expressed as the probability that the memory words in the system are of a certain word status type for the given period.

5. Lastly, the reliability of the memory system, which is the reliability of all the self-test periods including the current test period, is described.

4.3.1 Probability at Cell-Level (Bit-Level)

State Probability of Cell

See Section 4.1.2.

Transition Probability of Cell

Let $P_C^{H\to A}(t)$ as a probability that a healthy cell at time T is aged at time $T+t$ for any T. $P_C^{H\to F}(t)$ and $P_C^{A\to F}(t)$ $P_C^{H\to H}(t)$, $P_C^{A\to A}(t)$ and $P_C^{F\to F}(t)$ are defined

similarly.

$$P_C^{H\to H}(t) = e^{-(\lambda_1+\lambda_0)t} \tag{4.14}$$

$$P_C^{H\to A}(t) = \tfrac{\beta}{\alpha}\left(e^{-(\lambda_1+\lambda_0)t} - e^{-\lambda_2 t}\right) \tag{4.15}$$

$$P_C^{H\to F}(t) = 1 - \tfrac{1}{\alpha}\left(e^{-(\lambda_1+\lambda_0)t} - \beta e^{-\lambda_2 t}\right) \tag{4.16}$$

$$P_C^{A\to A}(t) = e^{-\lambda_2 t} \tag{4.17}$$

$$P_C^{A\to F}(t) = 1 - e^{-\lambda_2 t} \tag{4.18}$$

$$P_C^{F\to F}(t) = 1 \tag{4.19}$$

Probability of Soft Error in a Cell

A word is reliable if it has at most one memory cell error; memory cell has an error if it is faulty, or if it has a soft error. A soft error in memory cell is modeled using a Poisson distribution. Let $P_C^{0S}(t)$ and $P_C^{1S}(t)$ denote the probabilities of the absence and presence, respectively, of a soft error in a memory cell during $[T, T+t]$ for any T. They are expressed as:

$$P_C^{0S}(t) = e^{-\lambda_s t} \tag{4.20}$$

$$P_C^{1S}(t) = 1 - e^{-\lambda_s t} \tag{4.21}$$

4.3.2 Probability at Word-Level

Probability of Word

See Section 4.1.4.

Transition Probability of Word

Let $P_W^{H\to H+A}(t)$ denote a probability that an H word at time T is an H or A word at time $T+t$ for any T. $P_W^{H\to 1F0+1FA}(t)$, $P_W^{A\to A}(t)$, $P_W^{A\to 1F0+1FA}(t)$, $P_W^{1F0\to 1F0+1FA}(t)$, and $P_W^{1FA\to 1FA}(t)$ are defined similarly. Let $P_W^{A\to A}(t,i)$ denote a probability that an A word with i aged cells at time T is an A word at time $T+t$ for any T. $P_W^{A\to 1F0+1FA}(t,i)$ and $P_W^{1FA\to 1FA}(t,i)$ are defined similarly.

$$P_W^{H\to H+A}(t) = \left(P_C^{H\to A}(t) + P_C^{H\to H}(t)\right)^n \tag{4.22}$$

$$P_W^{H\to 1F0+1FA}(t) = \binom{n}{1} P_C^{H\to F}(t) \left(P_C^{H\to A}(t) + P_C^{H\to H}(t)\right)^{n-1} \tag{4.23}$$

$$P_W^{A}(t,i) = \binom{n}{1} \left(\frac{\beta}{\alpha}\right)^i \left(P_C^{H}(t)\right)^n \tag{4.24}$$

$$P_W^{A\to A}(t) = \textstyle\sum_{i=1}^{n} \frac{P_W^A(t,i)}{P_W^A(t)} P_W^{A\to A}(t,i) \tag{4.25}$$

$$P_W^{A\to A}(t,i) = \left(P_C^{A\to A}(t)\right)^i \left(P_C^{H\to A}(t) + P_C^{H\to H}(t)\right)^{n-i} \tag{4.26}$$

$$P_W^{A\to 1F0+1FA}(t) = \textstyle\sum_{i=1}^{n} \frac{P_W^A(t,i)}{P_W^A(t)} P_W^{A\to 1F0+1FA}(t,i) \tag{4.27}$$

$$\begin{aligned} P_W^{A\to 1F0+1FA}(t,i) = {} & \binom{i}{1} \left(P_C^{A\to F}(t)\right) \left(P_C^{A\to A}(t)\right)^{i-1} \\ & \cdot \left(P_C^{H\to A}(t) + P_C^{H\to H}(t)\right)^{n-i} + \left(P_C^{A\to A}(t)\right)^i \binom{n-i}{1} \left(P_C^{H\to F}(t)\right) \\ & \cdot \left(P_C^{H\to A}(t) + P_C^{H\to H}(t)\right)^{n-i-1} \end{aligned} \tag{4.28}$$

$$P_W^{1F0\to 1F0+1FA}(t) = \left(P_C^{H\to A}(t) + P_C^{H\to H}(t)\right)^{n-1} \tag{4.29}$$

$$P_W^{1FA}(t,i) = \binom{n}{1} \left(P_C^{F}(t)\right) \binom{n-1}{1} \left(\frac{\beta}{\alpha}\right)^i \left(P_C^{H}(t)\right)^{n-1} \tag{4.30}$$

$$P_W^{1FA\to 1FA}(t) = \textstyle\sum_{i=1}^{n-1} \frac{P_W^{1FA}(t,i)}{P_W^{1FA}(t)} P_W^{1FA\to 1FA}(t,i) \tag{4.31}$$

$$\begin{aligned} P_W^{1FA\to 1FA}(t,i) = {} & \\ & \left(P_C^{A\to A}(t)\right)^i \left(P_C^{H\to A}(t) + P_C^{H\to H}(t)\right)^{n-i-1} \end{aligned} \tag{4.32}$$

Probability of Soft Error in a Word

Let $P_W^{0S}(t)$ and $P_W^{1S}(t)$ denote the probabilities that no cell and that one cell has soft error in a word during $[T, T+t]$ for any T, respectively.

$$P_W^{0S}(t) = \left(P_C^{0S}(t)\right)^n \tag{4.33}$$

$$P_W^{1S}(t) = \binom{n}{1} P_C^{1S}(t) P_{n-1}^{0S}(t) \tag{4.34}$$

The probability $P_W^{\leq 1S}(t)$ of a word with at most one soft error during $[T, T+t]$ is expressed as:

$$P_W^{\leq 1S}(t) = \left(P_W^{0S}(t) + P_W^{1S}(t)\right)^v \cdot \left(P_W^{0S}(t_e) + P_W^{1S}(t_e)\right) \tag{4.35}$$

where $v = \lfloor t/t_s \rfloor$, which is the number of scrubbing intervals during $[0, t]$, and $t_e = t - vt_s$, which is the elapsed time after the last scrubbing.

We also use the probability that $n-1$ cells (or a word with exception of 1 cell) has no soft error during $[T, T+t]$ for any T. It is expressed as follows:

$$P_{n-1}^{0S}(t) = \left(P_C^{0S}(t)\right)^{n-1} \tag{4.36}$$

4.3.3 Reliability of a Memory Word

A memory word is reliable if it has at most 1 error at any given time T. Let $R_W^H(t)$ be a reliability of word, that is H at T, at $T+t$ for any T. $R_W^A(t)$, $R_W^{1FA}(t)$, and $R_W^{1FA}(t)$ are defined similarly.

$$R_W^H(t) = P_W^{H \to H+A}(t) \cdot P_W^{\leq 1S}(t) + P_W^{H \to 1F0+1FA}(t) \cdot P_{n-1}^{0S}(t) \tag{4.37}$$

$$R_W^A(t) = P_W^{A \to A}(t) \cdot P_W^{\leq 1S}(t) + P_W^{A \to 1F0+1FA}(t) \cdot P_{n-1}^{0S}(t) \tag{4.38}$$

$$R_W^{1F0}(t) = P_W^{1F \to 1F0+1FA}(t) \cdot P_{n-1}^{0S}(t) \tag{4.39}$$

$$R_W^{1FA}(t) = P_W^{1FA \to 1FA}(t) \cdot P_{n-1}^{0S}(t) \tag{4.40}$$

4.3.4 Reliability of Self-Test Period

The reliability of one self-test period is a probability that there are N words in H, A, $1F0$, or $1FA$ among $N+N_s$ words at the beginning of the period and they do not become $2F$ during the period. The reliability for the k-th period is given as

$$\begin{aligned} R_{period}(k,t) = {} & \frac{P_S^H(k)}{P_S^C(k)} \left(R_W^H(t)\right)^N \\ & + \sum_{i=1}^{N} \frac{P_S^A(k,i)}{P_S^C(k)} \left(R_W^H(t)\right)^{N-i} \left(R_W^A(t)\right)^i \\ & + \sum_{j=1}^{N} \sum_{i=0}^{N-j} \frac{P_S^{1F0}(k,i,j)}{P_S^C(k)} \left(R_W^H(t)\right)^{N-j-i} \left(R_W^A(t)\right)^i \left(R_W^{1F0}(t)\right)^j \\ & + \sum_{m=1}^{N} \sum_{j=0}^{N-m} \sum_{i=0}^{N-m-j} \frac{P_S^{1FA}(k,i,j,m)}{P_S^C(k)} \\ & \cdot \left(R_W^H(t)\right)^{N-m-j-i} \left(R_W^A(t)\right)^i \left(R_W^{1F0}(t)\right)^j \left(R_W^{1FA}(t)\right)^m \end{aligned} \tag{4.41}$$

Let $T_k = (k-1) \cdot t_t$ be the time at the beginning of a period k. $P_S^H(k)$ is a probability that a system of N words has only H words at the beginning of period k.

$$P_S^H(k) = \sum_{h=N}^{N+N_s} \binom{N+N_s}{h} \left(P_W^H(T_k)\right)^h \left(1 - P_W^H(T_k)\right)^{N+N_s-h} \tag{4.42}$$

$P_S^A(k)$ is a probability that a system of N words has H or A words including one or more A words at the beginning of period k. $P_S^{1F0}(k)$ and $P_S^{1FA}(k)$ are defined similarly.

$$P_S^A(k) = \sum_{i_A=1}^{N} P_S^A(k, i_A) \tag{4.43}$$

$$P_S^{1F0}(k) = \sum_{j_{1F0}=1}^{N} \sum_{i_A=0}^{N-j_{1F0}} P_S^{1F0}(k, i_A, j_{1F}) \tag{4.44}$$

$$P_S^{1FA}(k) = \sum_{m_{1FA}=1}^{N} \sum_{j_{1F0}=0}^{N-m_{1FA}} \sum_{i_A=0}^{N-j_{1F0}-m_{1FA}} P_S^{1FA}(k, i_A, j_{1F0}, m_{1FA}) \tag{4.45}$$

$P_S^A(k,i)$ is a probability that a system of N words has H or A words where the number of A words is i at the beginning of period k. $P_S^{1F0}(k,i,j)$ is defined similarly where the number of A words are $i(\geq 0)$ and $1F0$ words are $j(> 0)$.

$P_S^{1FA}(k,i,j,m)$ is defined similarly where the number of A words are $i(\geq 0)$, $1F0$ words are $j(\geq 0)$ and $1FA$ words are $m(> 0)$.

$$P_S^A(k,i) = \sum_{i=i_A}^{i_A+N_s} \binom{i_A+N_s}{i} \left(P_W^H(T_k)\right)^{N-i_A} \left(P_W^A(T_k)\right)^i \cdot \left(1 - P_W^H(T_k) - P_W^A(T_k)\right)^{i_A+N_s-i} \quad (4.46)$$

$$P_S^{1F0}(k,i,j) = \sum_{j=j_{1F0}}^{j_{1F0}+N_s} \binom{j_{1F0}+N_s}{j} \left(P_W^H(T_k)\right)^{N-i_A-j_{1F0}} \left(P_W^A(T_k)\right)^{i_A} \left(P_W^{1F0}(T_k)\right)^j \cdot \left(1 - P_W^H(T_k) - P_W^A(T_k) - P_W^{1F0}(T_k)\right)^{j_{1F0}+N_s-j} \quad (4.47)$$

$$P_S^{1FA}(k,i,j,m) = \sum_{m=m_{1FA}}^{m_{1FA}+N_s} \binom{m_{1FA}+N_s}{m} \left(P_W^H(T_k)\right)^{N-i_A-j_{1F0}-m_{1FA}} \cdot \left(P_W^A(T_k)\right)^{i_A} \left(P_W^{1F0}(T_k)\right)^{j_{1F0}} \left(P_W^{1FA}(T_k)\right)^{m_{1FA}} \cdot \left(1 - P_W^H(T_k) - P_W^A(T_k) - P_W^{1F0}(T_k) - P_W^{1FA}(T_k)\right)^{m_{1FA}+N_s-m} \quad (4.48)$$

$P_S^C(k)$ is the sum of $P_S^H(k)$, $P_S^A(k)$, $P_S^{1F0}(k)$, and $P_S^{1FA}(k)$.

$$P_S^C(k) = P_S^H(k) + P_S^A(k) + P_S^{1F0}(k) + P_S^{1FA}(k) \quad (4.49)$$

4.3.5 Reliability of Memory System

For memory to remain reliable, the system must utilize N words in H, A, $1F0$, or $1FA$ among $N + N_s$ words, and the system is assessed per self-test period. Thus, the reliability of the memory is the product of the reliabilities from the previous $(k-1)$-th self-test periods, and the reliability of the current period, and it is given as

$$R(t) = \left(\prod_{p=1}^{k-1} R_{period}(p, t_t)\right) \times R_{period}(k, t_e) \quad (4.50)$$

where t is in the k-th period, and $t_e\,(= t \bmod t_t)$ is the time elapsed after the last self-test.

Table 4.1: Baseline Memory Parameters

Parameters	Definition	Values
N	number of words	1e+5
N_s	number of spare words	50
n	number of bits per word	21
t_t	time bet. self-test & repair (hours)	240
t_s	time between scrubbing (hours)	0.1
λ_s	soft error rate (errors per hour)	10^{-7}
λ	rate from non-faulty to faulty	$10^{-11}, 10^{-10}, 10^{-9}$
λ_1	rate from healthy to aged	λ
λ_2	rate from aged to faulty	$(\alpha/\beta) \cdot \lambda$
λ_0	rate from healthy to faulty	$\lambda/(\alpha/\beta)$

4.4 Experimental Results

In this section, we present the evaluation results of the reliability model for the extended proposed strategy, and its hardware overhead.

4.4.1 Reliability Evaluation

The reliability is evaluated using a similar set-up from the previous chapter. Baseline parameters for this evaluation are shown in Table 4.1. The scrubbing interval t_s is 6 minutes, while BIST and aging test period t_t is 10 days. The reliability of the extended proposed strategy is compared to the reliability of the proposed strategy from Chapter 3 that does not consider aged words, as well as the reliability of the traditional repair [5,6] that only repairs uncorrectable words. Reliabilities are observed up to 50 years. The results show reliability evaluation for varying rates of λ.

Figure 4.8 shows the result for $\lambda = 10^{-11}$. The traditional method shows a downward trend at an early time. In Figure 4.9, given $\lambda = 10^{-10}$, the reliability of the proposed method is kept close to 1, while that of the previous method degrades after a few decades. In Figure 4.10, given $\lambda = 10^{-9}$, the reliability of the previous method has clearly degraded. This shows that when the degradation

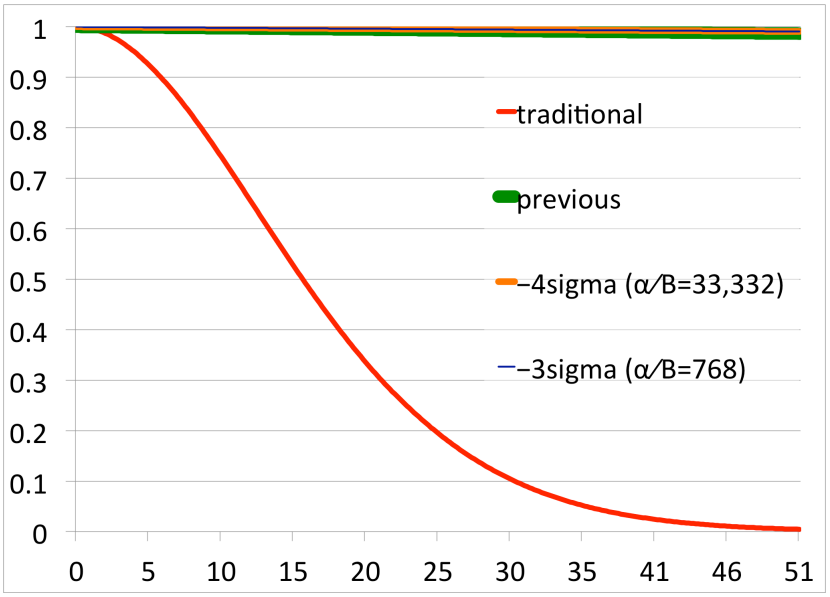

Figure 4.8: Reliability of extended proposed strategy vs previous vs traditional strategy for error rate of λ=10^{-11}

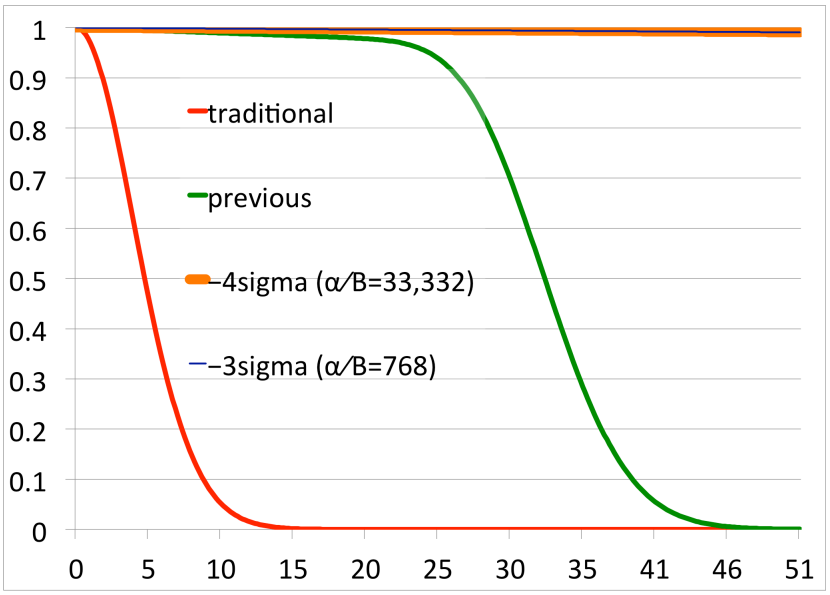

Figure 4.9: Reliability of extended proposed strategy vs previous vs traditional strategy for error rate of λ=10^{-10}

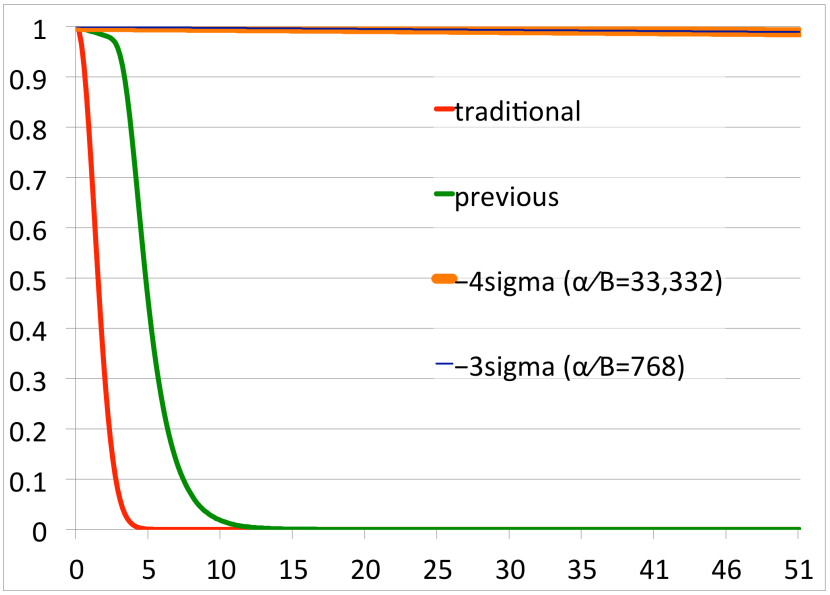

Figure 4.10: Reliability of extended proposed strategy vs previous vs traditional strategy for error rate of λ=10^{-9}

is not negligible, the proposed method is very effective compared to the other method.

4.4.2 Hardware Overhead

Area Overhead

The memory elements used for the extended proposed architecture is similar to that in Table 3.3, except for the extended diagnosis CAM, as shown in Table 4.2.

Table 4.2: Memory Elements Used for Extended Proposed Architecture

Memory Element	Number of Words	Word Size (in Bits)
Memory	1e+5	21
Spare	50	21
Remap CAM	50	34
Diagnosis CAM	256	59

Table 4.3: Hardware Overhead of Extended Remap Controller

Number of Words	Address Bit Count	Gate Count Previous [Table 3.4]	Gate Count Proposed	Gate Count Increase
8K+50 spare	14	1467	2858	1.95x
64K+50 spare	17	1570	3026	1.93x
1e+5+50 spare	17	1599	3056	1.91x
256K+50 spare	19	1663	3132	1.89x
512K+50 spare	20	1711	3182	1.86x

*K means exactly 1,024 bits

Each entry of the extended diagnosis CAM consists of the faulty address word, and two words, one of which being used for localizing faulty bits, and another being used for localizing aged bits as illustrated in Figure 4.4.

The remap controller of the extended in-field repair strategy using the Synopsys SAED90nm_EDK library is implemented. 5 controllers were implemented with increasing memory sizes. Table 4.3 shows the hardware overhead of 5 remap controllers. The numbers of gate count are expressed in 2-input NAND gate equivalents. The proposed controllers have less than 3,200 gates, a negligible quantity compared to the total number of gates for an SoC.

Power and Latency Overhead

For the aging-aware architecture, the power and latency overhead is similar to the previous architecture. The discussion for power overhead is similar to Section 3.5.2 since it uses the same estimation to perform the evaluation. For the latency overhead, it is almost similar except that the aging test needs an extra 2 write cycles to perform the test [17].

4.5 Chapter Summary

Self-test can now identify aged memory cells, along with faulty memory cells. Identification of aged cells enables development of a 3-state model for a memory cell which thus can be categorized as healthy, aged or faulty. It also allows to clas-

sify words into 5 status types for in-field repair. A novel strategy that combines aging-aware adaptive in-field self-repair and ECC, which enhances reliability further by repairing the classified faulty and aged words in order of vulnerability, is proposed. Repair of the more vulnerable word is guaranteed by cancelling previous repair of a less vulnerable word if necessary. Also, the remap controller's overhead is negligibly small. A stochastic reliability evaluation based on a Poisson distribution for transition rates between cell states, as well as soft errors, is formulated. The extended proposed strategy enhances memory reliability further and extends its lifetime by several decades compared to the previous methods.

5 Support for Memory of Higher Technology

Recent memory technologies are dramatically improving in a very high rate, which gives greater performance. Thus, memory can have certain features for a specific technology or may require optimized performance. The proposed architectures can be further adapted to higher technologies, and some key points are identified and described in detail in this chapter.

5.1 Memory with Burst Rate Operation

High speed memories have features such as burst rate operation. This means memory reads or writes are composed of one or more memory words. This tremendously speeds up time to access memory since the next set of words are just waiting to be read, in case of memory read, or the next set of words to be written are already included in a single access, in case of memory write.

The proposed architectures can be adapted to support burst rate operation. The implemented architectures presented in this work discuss word-level organizations. For such a case that memory accesses are composed of one or more memory words, the proposed strategies are applied in terms of memory blocks. One memory block is distinguished as a batch of one or more words. Thus, for a

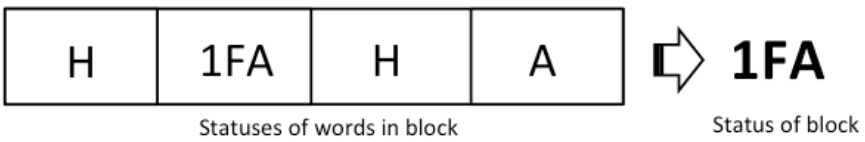

Figure 5.1: Block status for memory with burst rate operation

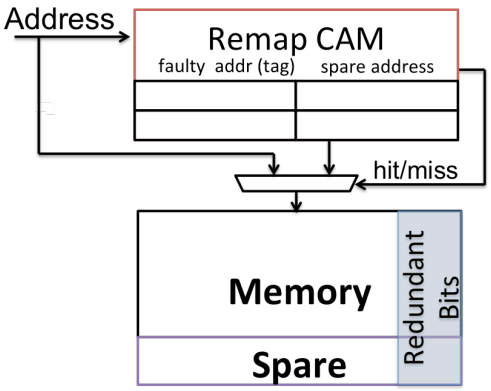

Figure 5.2: Remap CAM Access During User Mode

given burst rate, test and repair is done for each memory block, and the proposed strategies are still in effect. This also implies that a faulty block is repaired using a spare block. In this scenario, since a single block contains one or more words, the status type of the block depends on the worst word status type included in the said block. For example, a block is composed of 4 words as shown in as shown in Figure 5.1. It contains 2 healthy (H) words, 1 aged (A) word and an 1-faulty-aged (1FA) word. Since the block adopts the status of word with the worst status, the status of the block is 1FA. This block is repaired according to the strategy of its status.

Thus, the proposed architecture can be used for higher technology memories.

5.2 Latency Optimization

Without degradation of reliability, the proposed architecture can be further optimized in terms of performance. During user mode, the remap CAM can potentially cause latency issues since the memory address to be accessed is checked whether it is stored in the remap CAM or not as shown in Figure 5.2. In order to avoid this bottleneck, the architecture is optimized as shown in Figure 5.3. A memory address access is performed normally, while concurrently, the same address is searched in the remap CAM. Each address entry in the remap CAM is logically allocated to a spare address. A control signal indicates whether the memory address comes from memory or spare. The concurrency in the opera-

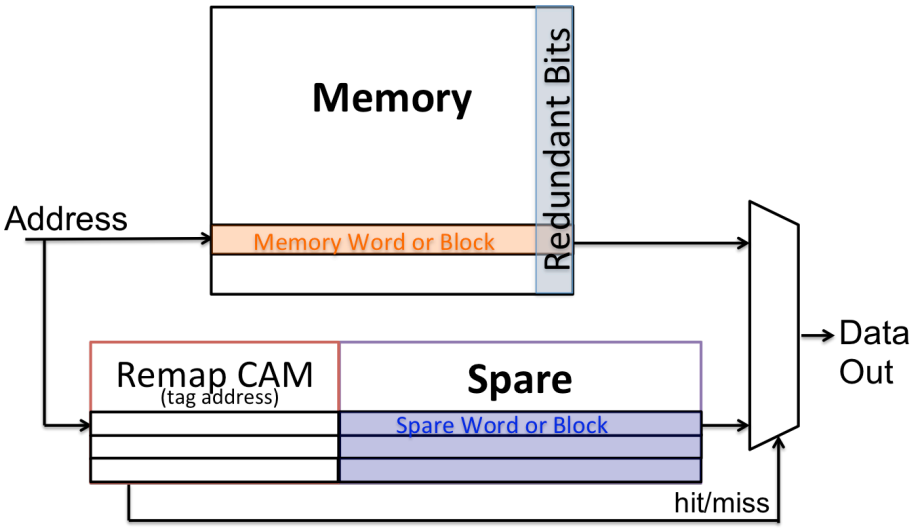

Figure 5.3: Modified architecture to optimize performance

tion of the normal memory access ensures that this output is just waiting for the control signal coming from the remap CAM, whether it is a hit or a miss. A miss indicates normal memory operation, while hit indicates access via the spare. Thus, the latency overhead only comes from time required to access the remap CAM, and the performance of the memory architecture is optimized.

5.3 Large Memory Block Size (up to 512 bits) and ECC Protection Strength

Higher memory technologies require memory access to be block-based as described in Section 5.1. Since each block is composed of multiple words, this implies block sizes to be large. The results from Figure 3.12 already describe the reliability for longer word lengths. However it only uses single error correction for the whole block, which is not enough to be properly protected. To be able to provide support for memory technologies like cache which requires block size of 512 bits (or 64 Bytes), stronger ECC is required to protect not just the whole block, but each memory word in the block as well. The work [22] experimentally shows how varying the number of bits protected improves performance of cache memory.

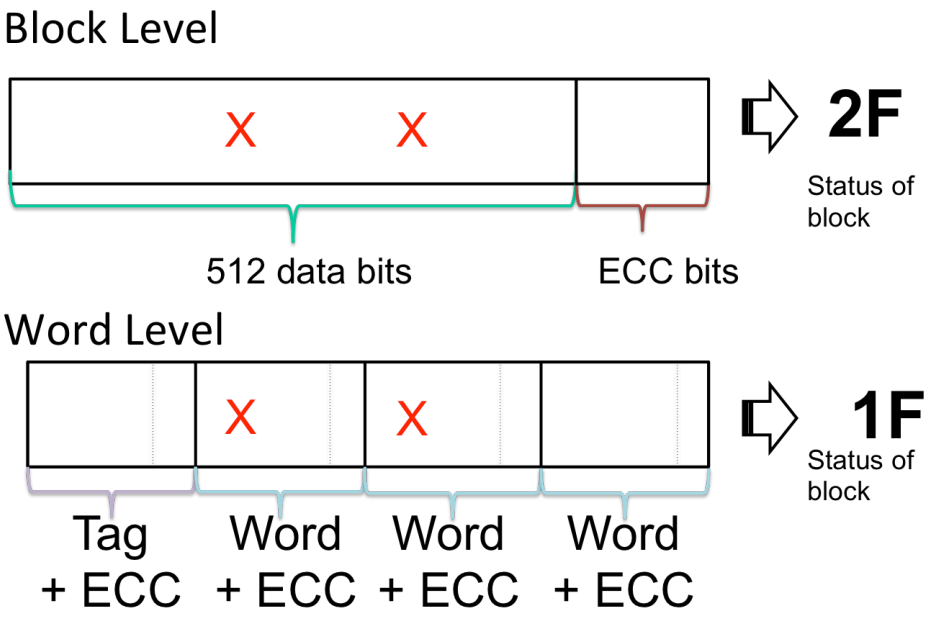

Figure 5.4: ECC for 512bit Block Size at Block and Word Level

The error correction schemes are described in terms of granularity, or the level of protection of a given number of bits. These include block level correction, word level correction, as well as stronger schemes, wherein the tag field of the cache is equally protected as well.

The proposed strategy can be adapted into large memory blocks as shown in Figure 5.4. For a 512bit block with only block level correction, the moment it suffers from 2 or more errors, its status is considered as uncorrectable (or 2F, as given from the strategy terminology for word or block status). For a 512bit block with word level correction, including protection for tag field, the status of the block depends only the worst status of the words among the block, such that even if there are 2 or more errors, as long as it can be protected in the word level, the block status is correctable (or 1F, as given from the strategy terminology for word or block status). To check how strengthening error protection in large memory block sizes affects reliability, the effect of granularity is adapted and several experiments are performed. The case where tag and data field uses same number of bits is considered. Cases for different data words such as 16-bit, 32-bit and 64-bit words are explored. A minimum base block size of 512 bits is used.

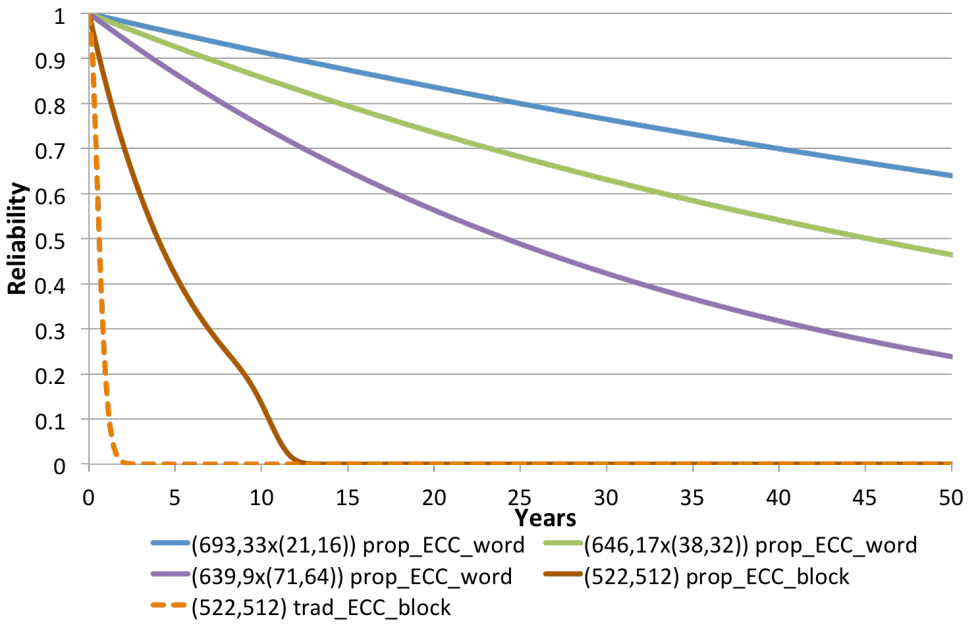

Figure 5.5: Reliability for 512bit Block Size Memory with ECC at Word Level

To illustrate the block size and word granularity, we cite an example for a 16-bit data word. For a 512 bit block, this fits 32 16-bit words. Taking into account the additional ECC bits, this corresponds to 32 21-bit codewords. Including an additional codeword for the tag field, this adds up to a total of 33 21-bit codewords for a total of 693 bits for each block. The corresponding block size calculation are considered for the 32-bit and 64-bit words accordingly. The results are shown in Figure 5.5, given that it uses the baseline parameters described in Table 3.2. Increasing the error protection increases the reliability. However, the increase of level of protection for smaller data word sizes correspond to increased block size. Also, since baseline parameters are used, increasing the number of spares will help improve reliability.

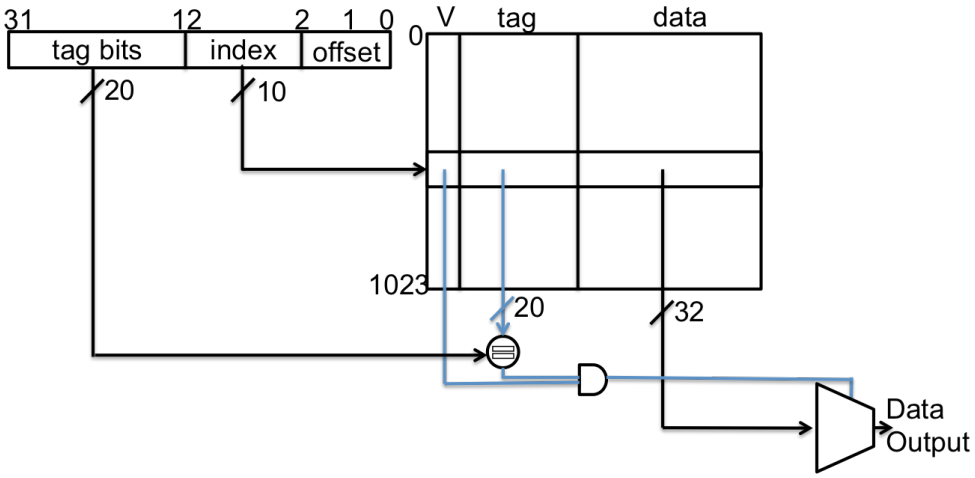

Figure 5.6: Direct-mapped Cache Memory

5.4 Implementation of Proposed Strategy to Cache Memory

The proposed architecture is expected to be adaptable into several high memory technologies. As an example, the concept of the how proposed architecture can be applied to cache memory is illustrated. A typical direct-mapped cache memory is used for the example as shown in Figure 5.6, wherein access to the cache makes use of a tag field to match if address is in the cache, and the corresponding data field. Given 32-bit addressing, the cache has 1024 entries, and has 10 index bits, 20 tag bits and 2 offset bits. Data is outputted as long as valid bit (V) is indicated to be set.

The proposed repair is implemented upon cache memory as shown in Figure 5.7. It follows the same way how the proposed strategy works wherein access is thru a CAM composed of tag and spares, when the cache entry is faulty. The tag is first searched in the CAM, then the corresponding data field contains the spare data. From which entry the access will come from is determined by the valid bit. A valid bit set to 1 at the cache entry indicates a normal access. A valid bit set to 1 at the remap CAM, and a valid bit set to 0 at the cache entry indicates a spare access. If valid bit is 0 for both cases, this implies that the memory address is not located in the cache and access to secondary memory is necessary. A similar

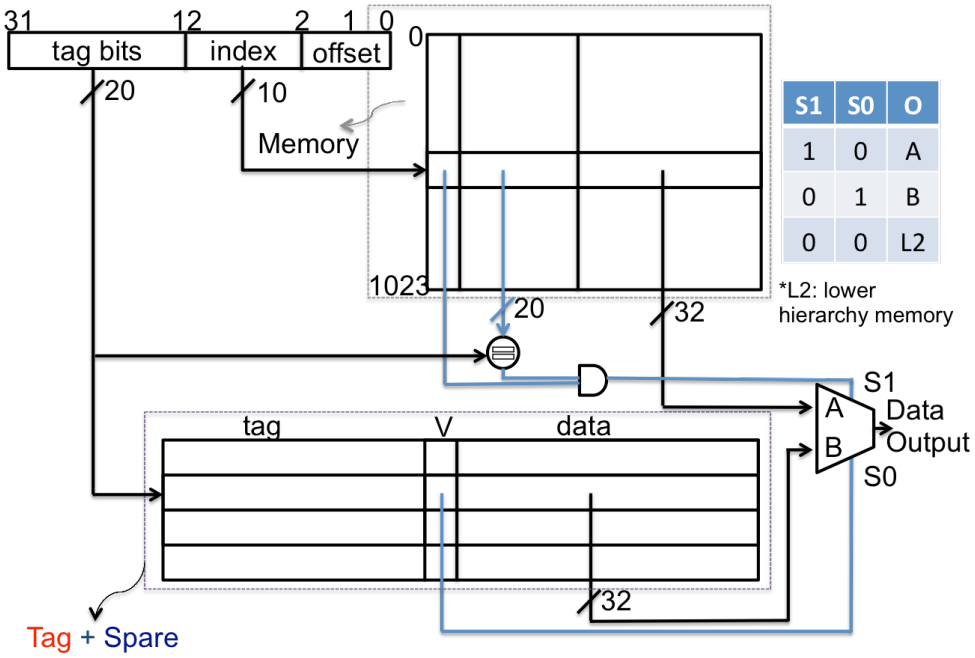

Figure 5.7: Proposed Repair Integrated into Cache

implementation can be used for set-associative cache but with separate allotment of spare for each set.

5.5 Summary

With the recent advancement of higher memory technologies, certain features for a specific technology or optimized performance are required for memory systems. Key points for specific features for high speed memories, such as burst operation, latency optimization, and large memory block sizes, have been addressed. These show that the proposed architectures can be adapted to fit these systems. An example of a cache implementation is also described.

6 Conclusion

6.1 Memory Architectures for Reliability Enhancement

The vulnerabilities encountered by embedded memory in SoCs have been presented, especially for reliability. The combination of self-repair and ECC is identified as a solution that can be efficiently used to perform memory repair. At the same time, a reliability problem that can occur from the combination of repair and ECC has been presented. Moreover, aging that can make memory more susceptible to failure has been explained. Two architectures are presented as solutions to enhance reliability.

The first architecture is the solution to the reliability problem of combination of repair and ECC. A novel in-field repair strategy that repairs uncorrectable words and possibly correctable words is proposed for ECC-based memory. An adaptive reconfiguring method is executed by the implemented remap controller to realize the proposed in-field repair strategy. A stochastic reliability model of the proposed strategy is established based on Poisson distributions for hard and soft errors. The proposed strategy enhances memory reliability in field compared to the traditional strategy, which repairs uncorrectable words only. Also, the hardware overhead is found to be small enough. These show that the proposed strategy extends memory lifetime in practical use.

The second architecture is an aging-aware implementation for ECC-based memory. Self-test can now identify aged memory cells, along with faulty memory cells. Identification of aged cells enables development of a 3-state model for a memory cell which thus can be categorized as healthy, aged or faulty. It also allows to classify words into 5 status types for in-field repair. A novel strategy that combines aging-aware adaptive in-field self-repair and ECC enhances reliability further by

repairing the classified faulty and aged words in order of vulnerability, is presented. Repair of the more vulnerable word is guaranteed by cancelling previous repair of a less vulnerable word if necessary. Also, hardware overhead is found to be small enough. A stochastic reliability evaluation based on a Poisson distribution for transition rates between cell states, as well as soft errors, is formulated. The extended proposed strategy enhances memory reliability further and extends its lifetime by several decades compared to the previous methods.

All in all, both architectures presented are solutions to the reliability problems that memory for SoCs encounter in field.

6.2 Summary

This thesis has presented reliability enhancement solution for memory in field. It identified the reliability problems that memory may encounter, and strategies and architectures that enhances reliability are proposed and presented. The combination of self-repair, ECC and aging test, when strategically utilized, enhances memory reliability further. The strategies can be adapted to memories of higher technology, and can be explored for future work.

References

[1] E. Marinissen, et al., "Challenges in embedded memory design and test," Proc. DATE'05, Vol. 2 pp. 722-727, Mar 2005.

[2] T. Wu, et al., "A memory yield improvement scheme combining built-in self-repair and error correction codes," ITC 2012, pp. 1-9, Nov 2012.

[3] M. Horiguchi, K. Itoh, **Nanoscale Memory Repair**, Springer, New York, NY, USA. 2011.

[4] M. Nicolaidis, et al., "A diversified memory built-in self-repair approach for nanotechnologies," Proc. IEEE VTS 2004, pp. 313-318, Apr 2004.

[5] M. Nicolaidis, and P. Papavramidou, "Transparent BIST for ECC-based memory repair," Proc. IEEE IOLTS 2013, pp. 216-213, Jul 2013.

[6] C.L. Su, et al., "An integrated ECC and redundancy repair scheme for memory reliability enhancement," IEEE DFT 2005, pp. 81-89, Oct 2005.

[7] K. Kang, et al., "NBTI induced performance degradation in logic and memory circuits: how effectively can we approach a relibility solution?," Proc. ASPDAC'08, Vol. 2 pp. 726-731, Jan 2008.

[8] S. Khan, S. Hamdioui, "Trends and challenges of SRAM reliability in nanoscale era," Proc. DTIS 2010, pp. 1-6, Mar 2010.

[9] Semico Research Corp. (2007, October 24). System(s)-on-a-chip - A Braver New World. Retrieved from http://www.semico.com/content/semico-systems-chip—braver-new-world

[10] W. Prates, et al., "Integrating embedded test infrastructures in SRAM cores to detect aging," Proc. IEEE IOLTS 2013, pp. 25-30, Jul 2013.

[11] J.F. Li, et. al., "A built-in self-repair scheme for semiconductor memories with 2-D redundancy," Proc. ITC 2003, pp. 393-402, Oct 2003.

[12] A. Benso, et al., "An on-line BIST RAM architecture with self-repair capabilities," IEEE Trans. on Reliability, 51(1):123-128, Mar 2002.

[13] A. Saleh, et al., "Reliability of scrubbing recovery-techniques for memory systems," IEEE Trans. on Reliability, 39(1):114-122, Apr 1990.

[14] B. Schroeder, et al., "DRAM errors in the wild: a large-scale field study," Proc. SIGMETRICS '09, pp. 193-204, Jun 2009.

[15] M. Nicolaidis. "Theory of transparent BISTs for RAMs," IEEE Trans. on Computer, Vol. 45, No. 10, October 1996.

[16] S. Bhardwaj, et al., "Predictive modeling of the NBTI effect for reliable design," Proc. IEEE CICC '06, pp. 189-192, Sept 2006.

[17] A. Ceratti, et al., "An On-Chip Sensor to Monitor NBTI Effects in SRAMs," Journal of Electronic Testing, Vol. 30, Issue 2, pp. 159-169, April 2014.

[18] P. Pouyan, et al., "Adaptive Proactive Reconfiguration: A Technique for Process-Variability-and Aging-Aware SRAM Cache Design," Trans. on VLSI Systems, pp. 1951-1955, Sept 2015.

[19] B. Jacob, S. Ng, and D. Wang, **Memory Systems: Cache, DRAM, Disk**, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[20] CACTI. Retrieved from http://quid.hpl.hp.com:9081/cacti/

[21] A.J. Van De Goor, et al., "Converting march tests for bit-oriented memories into tests for word-oriented memories," Proc. Int'l Workshop on MTDT 1998, pp.46-52, Aug 1998.

[22] N. Sadler and D. Sorin,"Choosing an Error Protection Scheme for Microprocessors L1 Data Cache," Proc. Intl Conference on Computer Design (ICCD) 2006, pp. 1-7, Oct. 2006.

[23] G. Mayuga, Y. Yamato, T. Yoneda, Y. Sato, and M. Inoue, "An ECC-based memory architecture with online self-repair capabilities for reliability enhancement," Proc. IEEE ETS 2015, pp. 1-6, May 2015.

[24] G. Mayuga, Y. Yamato, T. Yoneda, Y. Sato, and M. Inoue, "Reliability Enhancement of Embedded Memory with Combination of Aging-aware Adaptive In-Field Self-Repair and ECC," Proc. IEEE ETS 2016, pp. 1-6, May 2016.

[25] G. Mayuga, Y. Yamato, T. Yoneda, Y. Sato, and M. Inoue, "Reliability-Enhanced ECC-Based Memory Architecture Using In-Field Self-Repair," IEICE Transaction on Information and Systems, Vol. E99-D, No.10, Oct 2016.

Acknowledgements

I would like to express my utmost appreciation to the people who made this experience the most memorable and fruitful.

I would like to thank Prof. Michiko Inoue for the trust, support and guidance at every step in the way in this journey of becoming the best scientist that I can be.

I would like to thank Asst. Prof. Tomokazu Yoneda and Asst. Prof. Yuta Yamato for all their support on laboratory matters.

I would like to thank Prof. Yasuhiko Nakashima and Assoc. Prof. Fukuhito Ooshita for being part of my thesis commitee.

I would like to thank Prof. Yasuo Sato for providing his careful inspection and support on the technical aspects of my research.

I would like to thank all the previous and current laboratory members of Dependable System Laboratory for all their help and company during research meetings and laboratory events. I would like to thank all the administrative members of NAIST for all their help in official matters. I would like to thank the International Student Affairs Section for their tremendous help and countless efforts to international students in NAIST. I would like to thank the Filipino community in NAIST for their camaraderie. I would like to thank all the friends I have made in Japan for the wonderful experiences outside of the academic field.

Lastly, I would like to thank my family for their ultimate support and love.

Publication List

Peer-reviewed journal paper

1. Gian Mayuga, Yuta Yamato, Tomokazu Yoneda, Yasuo Sato, and Michiko Inoue. Reliability-Enhanced ECC-Based Memory Architecture Using In-Field Self-Repair, IEICE Transactions on Information and Systems, Vol. E99-D, No.10, Oct 2016. (Chapter 3)

Peer-reviewed international conferences

1. Gian Mayuga, Yuta Yamato, Tomokazu Yoneda, Yasuo Sato, and Michiko Inoue. Reliability Enhancement of Embedded Memory with Combination of Aging-aware Adaptive In-Field Self-Repair and ECC, In Proceedings of 21st IEEE European Test Symposium, pp. 1-2, Amsterdam, Netherlands, May 2016. (Chapter 4)

2. Gian Mayuga, Yuta Yamato, Tomokazu Yoneda, Yasuo Sato, and Michiko Inoue. An ECC-Based Memory Architecture with Online Self-Repair Capabilities for Reliability Enhancement, In Proceedings of 20th IEEE European Test Symposium, pp. 1-6, Cluj-Napoca, Romania, May 2015. (Chapter 3)

3. Gian Mayuga, Yuta Yamato, Tomokazu Yoneda, Yasuo Sato, and Michiko Inoue. An Online Repair Strategy and Reliability for ECC-Based Memory Architectures, In Proceedings of 15th IEEE Workshop on RTL and High Level Testing, pp. 1-6, Hangzhou, China, November 2014. (Chapter 3)

Technical report

1. Gian Mayuga, Yuta Yamato, Tomokazu Yoneda, Yasuo Sato, and Michiko Inoue. Reliability of ECC-based Memory Architecture with Online Self-repair Capabilities, IEICE Technical Report, Vol. 114, No. 384, DC2014-70, pp.19-24, December 2014. (Chapter 3)